CURSO 2008-09

SARA PROTOCOL FOR MANET

SECURE ADDRESS RANGE AUTOCONFIGURATION PROTOCOL FOR MANET

AUTORES:

VÍCTOR GARCÍA PÉREZ

FERNANDO GONZÁLEZ PEÑA

CARLOS MARTÍN SÁNCHEZ

PROFESOR:

LUIS JAVIER GARCÍA VILLALBA

PROYECTO DE SISTEMAS INFORMÁTICOS

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

Resumen

Las redes móviles ad hoc (MANET) permiten la comunicación de dispositivos móviles a través de enlaces inalámbricos. Debido a su naturaleza móvil, rutas multi-salto y a que no tienen una infraestructura, la configuración de este tipo de redes es un proceso complejo, ocupando actualmente un amplio campo de investigación.

Este trabajo propone un protocolo de autoconfiguración para redes MANET que asegura la unicidad de las direcciones IP y permite la entrada y salida de nodos de la red con una sobrecarga mínima y muy baja latencia.

La segunda parte del trabajo propone un mecanismo de seguridad para el protocolo basado en certificados digitales, ya que la seguridad en las redes MANET es un gran problema debido a los enlaces inalámbricos y su gestión generalmente distribuida.

Además, el trabajo incluye resultados obtenidos de distintas simulaciones del protocolo, realizadas en el simulador de redes NS-3.

Abstract

Mobile Ad Hoc Networks (MANET) make possible the communication between mobile devices through wireless links. Because of its mobile nature, multi-hop routes and the lack of infrastructure, the configuration of such networks is a complex process, being a wide area of research nowadays.

This work proposes an autoconfiguration protocol for MANET that ensures the uniqueness of the IP addresses and makes possible the entry and exit of nodes in the network with a minimum overhead and very low latency.

The second part of the work proposes a security mechanism for the protocol based on digital certificates, because the security in MANET is a big issue due to the wireless links and the commonly distributed management.

Moreover, the work includes results obtained from different simulations of the protocol, implemented in the network simulator NS-3.

Palabras clave

MANET, ad hoc, mobile, networks, autoconfiguración, protocolo, seguridad, encaminamiento, OLSR, criptosistema.

Keywords

MANET, ad hoc, mobile, networks, autoconfiguration, protocol, security, routing, OLSR, cryptosystem.

Víctor García Pérez Fernando González Peña Carlos Martín Sánchez

Lista de acrónimos

AA	Address Agent
AODV	Ad hoc On- Demand Distance Vector
APAC	Agent Based Passive Autoconfiguration for Large Scale manets
AROD	Address autoconfiguration with Address Reservation and Optimistic DAD for MANETs
ARP	Address Resolution Protocol
ARPANET	Advanced Research Projects Agency Network
BRP	Bordercast Resolution Protocol
CA	Certification Authority
CGSR	Cluster Gateway Switch Routing
CDMA	Code Division Multiple Access
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DAD	Duplicate Address Detection
DAD_REP	Detection Adress Detection REsPonse
DAD_REQ	Detection Adress Detection REQuest
DARPA	Defense Advanced Research Projects Agency
DHCP	Dynamic Host Configuration Protocol
DHCP-PD	Dynamic Host Configuration Protocol – Prefix Delegation
DNS	Domain Name Servers
DoS	Denial of Service
DSDV	Destination-Sequenced Distance-Vector

DS_REP	DNS Server REsPonse
DS_REQ	DNS Server REQuest
DSDV	Destination-Sequenced Distance-Vector Routing
DSR	Dynamic Source Routing
DYMO	Dynamic MANET On-demand
ECC	Elliptic curve cryptography
EMAP	Extensible Manet Autoconfiguration Protocol
FAMA	Floor Acquisition Multiple Access
FBCB2	Force XXI Battle Command, Brigade-and-Below
FDMA	Frequency Division Multiple Access
FFD	Full Funcion Device
FSRP	Fisheye State Routing Protocol
GASS	Grupo de Análisis, Seguridad y Sistemas
GC_REP	Gateway Control REsPonse
GC_REQ	Gateway Control REQuest
GNU	General Public License
GPS	Global Position System
GSM	Global System for Mobile Communications
HCQA	Hybrid Centralized Query-based Autoconfiguration
IANA	Internet Assigned Numbers Authority
IARP	IntrA-zone Routing Protocol
ICMP	Internet Control Message Protocol
IEEE	*Institute of Electrical and Electronics Engineers*

IERP	IntEr-zone Routing Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv6	Internet Protocol version 6
KMOV	Koyama Maurer Okamoto Vanstone
MAC	Media Access Control
MACA	Multiple Access with Collision Avoidance
MACAW	MACA with Acknowledgment
MACA-BI	MACA by Invitation
MANET	Mobile Ad Hoc Networks
MANET-WG	MANET Working Group
MD5	Message-Digest Algorithm 5
MPR	Multipoint Distribution Relay
MRTP	Multiflow Realtime Transport Protocol
NDP	Neighbor Discovery Protocol
NS3	Network Simulator 3
OLSR	Optimizad Link-State Routing
PACMAN	Passive Autoconfiguration for Mobile ad hoc Networks
PAN	Personal Area Networks
PDA	Personal Digital Assistant
PDAD	Passive Duplicate Address Detection
PKI	Public Key Infraestrucure
PPP	Point to Point Protocol

QoS	Quality of Service
RREP	Route REsPonse
RREQ	Route REQuest
RERR	Route Error
RF	Radio Frecuencia
RFC	Requests For Comments
RFD	Reduced Funcion Device
RIP	Routing Information Protocol
RSA	Rivest Shamir Adleman
SAODV	Secure Ad hoc On- Demand Distance Vector
SCTP	Stream Control Transfer Protocol
SDAD	Strong Duplicate Address Detection
SLAAC	StateLess Address AutoConfiguration
SURAN	Survivable Radio Network
TBRPF	Topology Dissemination Based on Reverse-Path Forwarding
TC	Topology Control
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TORA	Temporally Ordered Routing Algorithm
TPA	Transport Protocol for Ad hoc Networks
TSMA	Time Spread Multiple Access
TTL	Time To Live
UDP	User Datagram Protocol

USB	Universal Serial Bus
VoIP	Video over Internet Protocol
WDAD	Weak Duplicate Address Detection
WiFi	Wireless Fidelity
WRP	Wireless Routing Protocol
WSN	Wireless Sensor Networks
ZRP	Zone-Based Hierarchical Link State Routing Protocol

Índice

1. Introducción

Durante la realización de este proyecto, se ha trabajado en colaboración con el grupo de investigación GASS (Grupo de Análisis, Seguridad y Sistemas) de la Universidad Complutense de Madrid. El trabajo se enmarca dentro de los Proyectos de Avanza I+D TSI-020100-2008-365 y TSI-020100-2009-374, impulsados por el Ministerio de Ciencia y Tecnología. Estos proyectos consisten en el diseño e implementación de un esquema de seguridad y autoconfiguración segura de direcciones IP para redes MANET. Este esquema se puede dividir en tres módulos: encaminamiento, autoconfiguración y seguridad.

1.1. Contexto del trabajo

Internet se ha convertido en uno de los medios de difusión de información más importantes del mundo y las personas lo utilizan, en su vida cotidiana y profesional, como canal de comunicación, como sistema de colaboración masiva o simplemente como distracción. En los últimos años, la proliferación de dispositivos móviles cada vez más potentes (ordenadores portátiles, PDAs, móviles de última generación, etc.) posibilita la integración de éstos con Internet. Sin embargo, la característica de movilidad que acompaña a este tipo de dispositivos provoca el nacimiento de nuevos retos tecnológicos.

Una red móvil ad hoc o MANET (*Mobile Ad hoc NETwork*) es un conjunto de nodos móviles que se comunican entre sí a través de enlaces inalámbricos (*wireless*). Al contrario de las redes convencionales, una red MANET no necesita la existencia de una infraestructura previa ya que cada nodo se apoya en los demás para conseguir comunicarse con otro creando la llamada comunicación multi-salto.

Este tipo de redes tiene varios inconvenientes que una red convencional no presenta. La topología de este tipo de redes puede cambiar rápidamente y de una forma impredecible. Además, pueden surgir variaciones en las

capacidades de los nodos y enlaces, errores frecuentes en la transmisión y falta de seguridad.

También se deben tener en cuenta los recursos limitados de los nodos ya que normalmente una red ad hoc estará formada por dispositivos alimentados por una batería como sensores, teléfonos móviles, PDAs, etc.

Por último, cabe destacar uno de los principales inconvenientes de las redes MANET: su seguridad. Debido a distintas características de este tipo de redes, como los enlaces inalámbricos o la falta de infraestructura que pueda controlar y gestionar la red, es mucho más propensa que los otros tipos de redes a sufrir ataques.

1.2. Objetivos

El propósito inicial de este proyecto fue desarrollar un protocolo de autoconfiguración para redes MANET que ofreciese cierto nivel de seguridad. A partir de ahí, se definieron con más detalle los objetivos buscados.

Una primera necesidad que se debía resolver antes de poder iniciar el proyecto fue realizar un amplio trabajo de documentación del estado actual de los campos de investigación relacionados con el proyecto propuesto.

Tras conocer los diferentes protocolos de autoconfiguración y encaminamiento para redes MANET existentes, se decidió fijar como objetivo que el protocolo desarrollado debía garantizar que las direcciones IP asignadas automáticamente de forma distribuída en una red MANET sean únicas.

El protocolo debe cumplir esa premisa ofreciendo soluciones a los problemas inherentes a las redes MANET, como son la movilidad, falta de infraestructura, entradas y salidas imprevistas de nodos, etc.

Como segundo objetivo principal, se determinó que el protocolo no sólo debía ofrecer fiabilidad, sino también proporcionar cierta seguridad frente a ataques. El método elegido para intentar conseguir esa seguridad en un tipo de redes tan inseguras por definición fue la implementación de un módulo de

seguridad compuesto de una autoridad de registro y un criptosistema de clave pública.

El módulo de seguridad presentado en esta memoria se trata de una base que ofrece los servicios necesarios para desarrollar una infraestructura más amplia en un futuro. El objetivo que se fijó fue crear un entorno seguro al mismo tiempo que se crea la red MANET, que luego podría ser usado por otros módulos para efectuar tareas de gestión como monitorización, identificación, etc.

El último requisito que se impuso al proyecto fue realizar una implementación del protocolo propuesto en un entorno de simulación de redes. La implementación en código ejecutable se fue realizando paralelamente a la especificación teórica del protocolo presentado. Esto hizo posible realizar pruebas de las soluciones que iban surgiendo a lo largo del proyecto, por lo que el resultado final ha resultado ser no sólo robusto, sino eficiente.

Por supuesto, otra de las razones por las que se decidió implementar el protocolo fue que de esta forma se podría estudiar su comportamiento final y extraer conclusiones o comparativas con otros protocolos similares.

1.3. Estructura de la memoria

La presente memoria consta de tres grandes bloques:

BLOQUE I (capítulos 2 a 6): Se compone del trabajo de investigación y documentación realizado.

Primero se introducen en el capítulo 2 los conceptos importantes relativos a las redes inalámbricas.

A continuación se desarrolla en el capítulo 3 la necesidad de protocolos de autoconfiguración en las redes MANET, y el estado del arte de este campo de investigación.

En el capítulo 4 se hace un estudio de los diferentes protocolos de encaminamiento en redes MANET propuestos hasta la fecha. Se realiza además una comparativa entre los diferentes tipos de protocolos, y también se exponen las razones por las que se eligió uno de ellos como base para el protocolo de autoconfiguración propuesto en esta memoria. Seguidamente en el capítulo 5 se explica este protocolo en profundidad.

Para finalizar con este bloque, en el capítulo 6 se desarrollan los conceptos necesarios para entender el tipo de seguridad que necesitan las redes MANET, y se hace un repaso a los diferentes mecanismos de seguridad existentes.

BLOQUE II (capítulos 7 a 9): En él se presenta la contribución realizada. Se trata de un protocolo de autoconfiguración para redes MANET, que busca ofrecer fiabilidad y seguridad.

El punto 7 contiene una exposición del protocolo del que se parte, explicando las principales características y funcionalidades.

El capítulo 8 se centra en el nuevo protocolo, SARA. En él se presentan los problemas a los que intenta dar solución, y se detallan las nuevas ideas introducidas. Para terminar, se desarrolla una definición formal en profundidad del nuevo protocolo.

El capítulo 9 recoge el trabajo realizado para crear un entorno seguro en las redes MANET: el módulo de seguridad aplicado en el protocolo SARA. Este módulo se compone básicamente en la implementación de un criptosistema de clave pública que permite cifrar y firmar mensajes. Debido al tipo de redes en el que se usará el protocolo SARA, el módulo de seguridad sigue unas pautas explicadas con detalle en su capítulo correspondiente.

BLOQUE III (capítulos 10 a 12): conclusiones, referencias y anexos. En el capítulo 10 discutimos los resultados obtenidos, e introducimos las ideas que se podrían desarrollar en futuras investigaciones en torno a la presente memoria.

El capítulo 11 contiene una lista con las referencias a artículos, libros, y publicaciones usadas en la memoria.

En el capítulo 12 se han incluido como anexos algunas cuestiones que no tenían cabida en el resto de la memoria. Contienen detalles concretos que quedan a disposición de quien quiera conocer en profundidad alguno de los temas tratados.

BLOQUE I

2. Redes inalámbricas

Las redes inalámbricas han tenido un gran impacto a nivel mundial desde la segunda guerra mundial, donde se empezó a usar para estrategias militares. Desde entonces, estas redes se han seguido desarrollando y sus usos han crecido significativamente; por ejemplo, en tecnologías como los teléfonos móviles que son parte de una gran red inalámbrica guiada por satélites, grandes áreas *Wifi*, ofrecidas en diferentes puntos de ciudades, conexiones *Bluetooth* entre dispositivos, el sistema *GPS* o simplemente una red local de una casa o una empresa.

Uno de los grandes factores de crecimiento de este tipo de redes, radica en que las redes tradicionales no permiten la movilidad de sus usuarios. En cambio, las redes inalámbricas permiten el movimiento libre dentro de la red evitando muchos problemas de cableado, dando mucha más libertad a sus usuarios y reduciendo mucho los costes de mantenimiento.

En esta sección se presentan las principales características de de las redes inalámbricas así como los diferentes tipos de redes existentes. Después se detalla un tipo de estas redes, las redes ad hoc.

2.1 – Características de las redes inalámbricas

Las redes inalámbricas poseen características propias que las diferencian de las redes cableadas:

- **Topologías dinámicas:** los nodos pueden moverse libremente en direcciones arbitrarias y con distinta velocidad.

- **Ancho de banda restringido:** restricciones impuestas por el canal inalámbrico, tal como acceso múltiple, interferencias, ruido, disponibilidad limitada del espectro, etc.

- **Energía limitada:** hay casos en los que la energía en las estaciones es limitada, por ejemplo en las redes de sensores.

- **Seguridad física limitada:** son susceptibles de tener carencia de seguridad por su conexión inalámbrica y pueden ser atacadas fácilmente.

Estas características, suponen una serie de ventajas e inconvenientes:

Ventajas:

- Permiten el movimiento de los usuarios por la red.

- Al ser no cableadas, el acceso a la red es mucho más cómodo, ya que cualquiera que este dentro del alcance de la red es capaz de conectarse a ella.

- Nos ahorra mucho gasto en infraestructura, evitando los cables y las obras que puede conllevar incluirlos en una oficina o casa.

- El número de usuarios que se puedan conectar a la red, no depende del número de cables físicos que se tengan, cualquiera podrá meterse en ella sin necesidad de ninguna infraestructura previa.

Desventajas:

- Pérdida de velocidad en las transmisiones, debido a las interferencias y pérdidas de señal que el ambiente pueda crear.

- El principal problema de estas redes es la seguridad, ya que los datos viajan por el aire y cualquiera puede introducirse en la red para leer los paquetes enviados a través de esta o para realizar comportamientos maliciosos que puedan perjudicar el funcionamiento de la red o de otros usuarios. Para solucionar esto, es necesario un mecanismo de seguridad, el

cual añadirá sobrecarga a la red y un nivel más alto de cómputo en los nodos.

2.2. Tipos de redes inalámbricas

Desde que nacieron las redes inalámbricas, se han realizado distintas clasificaciones: por su alcance, topología, infraestructura, número de saltos, etc. pero esta sección se centra en la clasificación por infraestructura, para luego dar paso a la siguiente sección, que trata en detalle un tipo de redes partiendo de esta clasificación:

- **Redes con infraestructura:** Necesitan de una entidad que actúe como servidor o punto de acceso, y sea quien gestione la red.

- **Redes sin infraestructura (ad hoc):** no tienen una entidad central, por lo que la administración de la red debe ser distribuida, todos los nodos pueden actuar como *routers* y es necesaria la comunicación entre ellos en modo multi-salto. Se necesita, por tanto, un mecanismo de autoconfiguración.

2.3 Redes Ad Hoc:

2.3.1. Introducción

Una red móvil ad-hoc o MANET (*Mobile Ad Hoc Network*) es una colección de nodos móviles autónomos que se comunican entre si mediante enlaces inalámbricos, dónde no existe una infraestructura de red fija y la administración se realiza de forma descentralizada. En este entorno, los nodos cooperan para llevar a cabo tareas fundamentales para el correcto funcionamiento de la red, tales como la configuración de la red, el mantenimiento y el encaminamiento. Esta cooperación permite la comunicación entre nodos no conectados directamente, enviando los paquetes a través de caminos multi-salto, en los que los nodos intermedios retransmiten el paquete hasta que alcanza el destino.

Este tipo de redes tienen una larga lista de características que han despertado un importante interés en los investigadores durante los últimos años. Entre estas características se encuentra su bajo coste, la facilidad a la hora de crear una red puesto que no requieren infraestructuras, la flexibilidad y el soporte de movilidad. Estas características se amplían en la siguiente sección. Sin embargo, algunas de las características que convierten a las MANETs en una solución interesante en determinados escenarios, imponen una serie de restricciones que dificultan su diseño e implementación.

En primer lugar, el empleo de nodos autónomos implica una escasez de recursos, tanto a nivel computacional como energético. Además, la movilidad de los nodos da lugar a una topología dinámica de la red, por lo que el período de validez de las rutas suele ser corto. La utilización de un medio compartido para la transmisión de los paquetes eleva el índice de colisiones, ancho de banda limitado y conlleva varios problemas de seguridad. Entre estos problemas cabría destacar la escucha de paquetes y la negación de servicio.

En general, cualquier propuesta real aplicable a una MANET deberá tener en cuenta las restricciones impuestas por estas características inherentes a este tipo de redes. Hasta ahora, los esfuerzos de investigación, guiados por el grupo de trabajo MANET dentro del IETF, se han centrado, principalmente, en temas relacionados con el encaminamiento. Sin embargo, existen otros aspectos no menos importantes que deben ser abordados en el diseño de una red de elevada funcionalidad y disponibilidad.

2.3.2 – Evolución histórica

El inicio de la historia de las redes se establece a mediados de los 60 cuando los militares crearon proyectos de defensa como DARPA estableciendo ARPANET.

Pero no es hasta principios de 1970 cuando en la universidad de Hawai se crea el primer proyecto a gran escala usando la radio para la transmisión de información, el proyecto se conocerá ampliamente como ALOHA.

En base al trabajo realizado en Hawai se llegó a construir una arquitectura distribuida llamada PARNET, financiada por DARPA en 1972. PARNET permitía la comunicación de usuarios móviles entre grandes áreas geográficas, compartiendo el mismo ancho de banda y evitando los efectos de los caminos múltiples.

La tecnología avanzó tan rápidamente en los 70, que se crearon varios sistemas de comunicación móvil como los teléfonos celulares, los sistemas de radio búsqueda, satélites, etc. Se estaban empezando a asentar las bases de las redes móviles ad hoc.

Posteriormente, en 1983, DARPA crea el proyecto SURAN, que se encargó de tratar el área de escalabilidad, seguridad, procesamiento y gestión de energía. Se desarrollaron dispositivos de bajo coste que permitían soportar los algoritmos de encaminamiento, escalar a miles de nodos las redes y soportar ataques de seguridad.

A mitad de los 90, se produce el auge de las redes inalámbricas con la llegada de las tarjetas de radio 802.11 a los ordenadores personales y portátiles.

Freebersyser en 2001[1] y Jain en 2003[2], proponen en sus artículos la idea de una red con una infraestructura mínima, a este tipo de redes, la IEEE le dio el nombre de “redes ad hoc”.

El establecimiento del estándar IEEE 802.11, permitió el rápido desarrollo de esta tecnología no solo en el ámbito militar, también en el ámbito comercial.

Desde entonces y hasta ahora, se están desarrollando e investigando nuevos protocolos tanto de autoconfiguración como de encaminamiento, así como sistemas de seguridad que aseguran tanto la integridad de los mensajes de la red como la autenticación de sus nodos.

2.3.3. Características

Los nodos que forman la red están equipados con transmisores y receptores inalámbricos que usan antenas que pueden ser omnidireccionales (*broadcast*), dirigidas (punto a punto), dirigibles o alguna combinación de éstas. En un determinado instante de tiempo, dependiendo de la posición de los nodos y de los rangos de cobertura de sus receptores y transmisores, de su potencia de transmisión y los niveles de interferencias, existe entre los nodos una conectividad en forma de un grafo aleatorio multi-salto o red “ad hoc”. Esta topología ad hoc puede cambiar con el tiempo debido al movimiento de los nodos o al cambio de los parámetros de recepción o de transmisión.

Las MANETs tienen varias características destacables:

- **Topología dinámica:** Los nodos se mueven arbitrariamente. De esta forma, la topología de la red puede cambiar aleatoriamente de forma rápida e impredecible, afectando tanto a enlaces bidireccionales como unidireccionales.

- **Limitaciones de ancho de banda:** variabilidad en la capacidad de los enlaces. Los enlaces inalámbricos continúan teniendo una capacidad sensiblemente más baja que los cableados. Además, el rendimiento de las comunicaciones inalámbricas suele ser mucho menor que el máximo previsto, debito a la multitud de factores que lo degradan (ruido, acceso múltiple, atenuación de la señal). Una consecuencia directa de la limitada capacidad de los enlaces es que la congestión se convierte en un fenómeno habitual y deja de ser algo ocasional. La demanda de ancho de banda por parte de las aplicaciones se aproximará o superará frecuentemente la capacidad de la red. Tanto si la red MANET es una red aislada como si es una extensión de un red fija, sus usuarios demandarán los mismos servicios. Estas demandas continuarán aumentando debido a la expansión de las aplicaciones multimedia y el desarrollo de las aplicaciones distribuidas.

- **Limitaciones de consumo energético:** Algunos o incluso todos los nodos que forman la MANET están alimentados por baterías u otros mecanismos de capacidad limitada. Para estos dispositivos la optimización del consumo energético es una cuestión primordial.

- **Limitada seguridad física:** Las redes móviles inalámbricas están, por lo general, más expuestas a problemas de seguridad que las redes físicas. El incremento de la probabilidad de sufrir escuchas, suplantación de identidad o denegación de servicio debe ser tenido en cuenta. Las técnicas existentes de seguridad a nivel de enlace aplicadas a redes inalámbricas reducen los riesgos de seguridad. Un beneficio de la naturaleza descentralizada del control de red en MANETs proporciona una robustez adicional ante los fallos en ciertos puntos aislados de la red que enfoques más centralizados.

Además, algunas de las posibles redes (redes militares, redes de carreteras) pueden ser relativamente grandes. La necesidad de escalabilidad no es exclusiva de las MANETs. Sin embargo, en el caso de estas redes es una cuestión de vital importancia puesto que si pretenden ser flexibles y facilitar la entrada de nuevos dispositivos, la escalabilidad no debe ser un problema. La dificultad para solucionar este problema reside en las características mencionadas anteriormente, que dificultan la operación con grupos con un elevado número de nodos.

2.3.4. Arquitectura

Al igual que en los demás tipos de redes, los mecanismos que proporcionan los servicios de red se distribuyen en capas, formando una pila. Se describe a continuación el modo de operación de cada una de las capas.

I. Capa Física

Los nodos pertenecientes a una red inalámbrica se comunican entre si utilizando como medio de transmisión el espacio libre, es decir, se comunican mediante canales de radiofrecuencia (RF). Los canales de radiofrecuencias presentan características que dificultad la calidad de servicio en redes

inalámbricas, algunas de los efectos que sufre la señal cuando se transmite en un canal RF son: el efecto *doopler*, la atenuación de la señal, el desvanecimiento por multitrayecto, etc. A la fecha, lo más común, es que los nodos de las redes MANET cuenten con antenas omnidireccionales que les permite comunicarse con cualquier otro dispositivo dentro de su rango de cobertura, el uso de este tipo de antenas influye en la velocidad de transmisión de las redes MANET. Cuando un nodo desea transmitir o recibir, los nodos vecinos deben estar en silencio haciendo que la capacidad de la red no se ocupe al 100%. Una de las soluciones propuestas en la literatura para solventar el problema de nodos vecinos es el uso de antenas unidireccionales.

Los estándares IEEE 802.11x [1] definen interfaces para canales de RF en las bandas de los 2.4GHz y de los 5GHz, siendo la primera la más extendida. Esta banda está muy saturada debido a que también es utilizada por otro tipo de dispositivos y redes como lo son, los hornos de microondas y las redes Bluetooth.

Todo lo anterior da como resultado que los canales de RF utilizados por las redes MANET son poco fiables.

II. Capa MAC

En general, existen básicamente dos categorías principales de protocolos de control de acceso al medio: los protocolos de acceso aleatorio, en los cuales los nodos compiten entre sí para ganar el acceso al medio de transmisión compartido, y los protocolos de acceso controlado, en los cuales un nodo maestro o de infraestructura decide cuál de los nodos puede acceder al medio de transmisión en cada momento. La falta de una infraestructura y la naturaleza *peer-to-peer* de las redes MANET hacen que los protocolos de acceso aleatorio sean la opción natural para el control del acceso al medio en redes MANET.

Algunos ejemplos de los protocolos MAC utilizados en las redes MANET son los siguientes: MACA (*Multiple Access with Collision Avoidance*), MACAW (*MACA with Acknowledgment*), MACA-BI (*MACA by Invitation*) y FAMA (*Floor Acquisition Multiple Access*). De entre las diversas propuestas, el Comité IEEE

802.11 eligió a CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance), una variante de MACA, como la base para sus estándares, debido a su inherente flexibilidad y porque resuelve los problemas de los terminales oculto y expuesto a través del sencillo mecanismo de intercambio de señales de control RTS/CTSDATA/ACK.

Entre los Protocolos MAC de Acceso Controlado se encuentran los siguientes: TDMA (*Time Division Multiple Access*), FDMA (*Frequency Division Multiple Access*), CDMA (*Code Division Multiple Access*) y TSMA (*Time Spread Multiple Access*).

Aunque estos protocolos raramente se utilizan en redes MANET, se prefieren en ambientes que necesitan garantías de Calidad de Servicio (QoS), ya que sus transmisiones están libres de colisiones. Su aplicación está principalmente adaptada a redes como *Bluetooth* y a otras redes MANET basadas en grupos (clusters) de nodos, en las cuales el acceso al medio está controlado por nodos maestros que deciden qué nodo del grupo es el que tiene acceso al canal, posibilitando así transmisiones libres de contienda y colisión. Entre las propuestas para mejorar el comportamiento de los protocolos MAC para redes MANET basadas en protocolos de acceso aleatorio se incluyen, por ejemplo, algoritmos para reducir el consumo de energía de los nodos móviles que permiten que los nodos “duerman” (esto es, que pasen a un estado de bajo consumo de energía) durante el periodo ocioso; diversos algoritmos de gestión de paquetes; reducción de los radios de transmisión y de recepción; ajuste de la velocidad de transmisión entre nodos vecinos dependiendo de la distancia que los separa, y diferentes esquemas de codificación y modulación de señales.

III. Capa de Red

Una característica especialmente importante de los protocolos de encaminamiento para redes MANET es que deben poder adaptarse rápidamente a lo cambios continuos de la red, con el fin de mantener las rutas entre los nodos que se están comunicando. De manera general, los protocolos de encaminamiento para redes MANET se clasifican en dos categorías

principales: proactivos y reactivos, aunque también existen otras clasificaciones. Los protocolos proactivos mantienen tablas que almacenan la información de encaminamiento y periódicamente, o ante cualquier cambio en la topología de la red, disparan un mecanismo de propagación de actualización a través de la red, con el fin de mantener una idea real del estado de la red. Esto puede causar una cantidad importante de paquetes de señalización (sobrecarga) que afecte la utilización del ancho de banda, el caudal (*throughput*), así como el consumo de energía. La ventaja es que las rutas a cada destino están siempre disponibles sin el incremento de paquetes de señalización que ocasiona un mecanismo de descubrimiento de rutas, pero tales protocolos tienen problemas para funcionar adecuadamente cuando en la red se presenta una alta tasa de movilidad o cuando hay un gran número de nodos en la red. Ejemplos de protocolos de esta categoría son: DSDV *(Destination-Sequenced Distance-Vector*), WRP (*Wireless Routing Protocol*), CGSR (*Cluster Gateway Switch Routing*), FSRP (*Fisheye State Routing Protocol*), OLSR (*Optimizad Link-State Routing*) y TBRPF (*Topology Dissemination Based on Reverse-Path Forwarding*). Los últimos dos protocolos preactivos se han convertido en RFC´s (*Requests For Comments*) experimentales aceptados por la IETF.

Por otro lado, los protocolos de encaminamiento por demanda o reactivos se caracterizan por iniciar un mecanismo de descubrimiento de ruta cuando una fuente necesita comunicarse con un destino al cual no sabe cómo llegar. El descubrimiento de ruta se realiza normalmente mediante una inundación de la petición. De manera general, el encaminamiento por demanda requiere menos sobrecarga que el encaminamiento basado en tablas (proactivo), pero incurre en un retraso de descubrimiento de ruta cada vez que se requiere un nuevo camino.

Las diferencias entre los protocolos por demanda están en la implementación del mecanismo de descubrimiento de ruta y en las optimizaciones del mismo. Ejemplos de protocolos reactivos son los que se mencionan a continuación: El protocolo DSR (*Dynamic Source Routing*) que está muy cerca de convertirse en RFC experimental. Por su lado, el protocolo AODV (*Ad hoc On- Demand Distance Vector*) ya es un RFC experimental. Otro

protocolo reactivo que inició el camino de convertirse en RFC pero que ya no ha tenido avances en ese proceso es TORA (*Temporally Ordered Routing Algorithm*). El protocolo de encaminamiento DYMO (*Dynamic MANET On-demand*) es el último protocolo reactivo unicast que está siendo considerado por el MANET-WG (*MANET Working Group*) para convertirse en RFC, pero se encuentra apenas en sus primeras fases para lograrlo.

Además de los protocolos proactivos y reactivos están los protocolos híbridos. El protocolo ZRP (*Zone-Based Hierarchical Link State Routing Protocol*) es un ejemplo de protocolo híbrido que combina los enfoques proactivo y reactivo, tratando de combinar las ventajas de ambos.

IV. Capa de transporte.

Las redes MANET tienen como objetivo final la implantación de la Internet inalámbrica omnipresente y ubicua, por lo que está basada en la pila de protocolos TCP/IP. Esto ha significado que prácticamente todos los esfuerzos de investigación y desarrollo del MANET-WG estén principalmente dirigidos al desarrollo de algoritmos de encaminamiento especialmente adecuados para este tipo de redes, sin que los demás protocolos de la pila se vean afectados. Aun así, en los últimos años se han presentado varias propuestas encaminadas a mejorar las prestaciones de los principales protocolos de transporte de la Internet, TCP (*Transmisión Control Protocol*) y UDP (*User Datagram Protocol*), en entornos MANET. Entre estas mejoras está el emplear mecanismos de señalización explícita que permitan tener un TCP “amigable” con las redes MANET, para que no active, erróneamente, el mecanismo de control de congestión ante la pérdida de la ruta entre el nodo fuente y el destino final debida a un fallo en el enlace. Otros autores han propuesto la creación de TPA (*Transport Protocol for Ad hoc Networks*), un nuevo protocolo de transporte especialmente diseñado para entornos ad hoc, que incluye mecanismos para detección de pérdida del enlace y recuperación de ruta, además de que establece un mecanismo de control de congestión diferente al de TCP. Los protocolos de transporte SCTP (*Stream Control Transfer Protocol*) y el MRTP (*Multiflow Realtime Transport Protocol*) han sido especialmente

diseñados para ofrecer un mejor transporte de datos a las aplicaciones de tipo media-streaming y de tiempo real.

V. Capa de aplicación.

Uno de los objetivos del MANET-WG es que las mismas aplicaciones diseñadas para la Internet actual puedan funcionar en las MANETs. Esto es válido actualmente para las aplicaciones de transmisión de datos que pueden conformarse con una entrega de datos bajo la filosofía de Best Effort, sin grandes requerimientos de ancho de banda ni restricciones en cuanto al tiempo de entrega de los paquetes. Sin embargo, Internet también se utiliza actualmente para otros tipos de aplicaciones, como aquellas que permiten el acceso en tiempo real a contenidos de audio y vídeo en repositorios remotos que producen flujos continuos de datos (*streams*), así como aplicaciones multimedia interactivas de tiempo real, como la voz sobre IP (*Video over Internet Protocol*, VoIP) o telefonía IP (*IP Telephony*), la vídeo conferencia y la vídeo telefonía, que demandan a la red de transporte ciertas garantías mínimas de retardo en la entrega de paquetes, de variación de este retardo y de ancho de banda, pero que por otro lado pueden soportar cierto nivel de pérdida de paquetes, sin que por tanto la información se vuelva inútil.

Debido a que el ancho de banda y la fiabilidad de los enlaces de las redes MANET es mucho menor que en las redes cableadas, hay muchos retos por resolver para que este tipo de aplicaciones puedan ser soportadas de manera adecuada en los entornos ad hoc. Algunas propuestas van en el sentido de disminuir lo más posible el ancho de banda requerido por las aplicaciones, las cuales se basan en nuevas técnicas de codificación y compresión de la información, sacrificando la calidad de la información para todos los nodos y condiciones de la red por igual. Otra dirección que se está considerando cada vez más, es hacer que las aplicaciones mismas se adapten a las condiciones dinámicas de las redes MANET y a las capacidades particulares de los nodos comunicantes en ambos extremos.

2.3.5. Aplicaciones

Es fácil encontrar situaciones donde se ve la utilidad de las redes MANET. Uno de los ejemplos más clásicos (aunque también discutido) es una reunión de trabajo: un grupo de personas con ordenadores portátiles o PDAs. Son de distintas empresas y por tanto sus direcciones son distintas. Tal vez en la sala haya acceso a Internet y puedan usar por ejemplo IP móvil, pero ¿para qué pasear sus datagramas por todas la ciudad o todo el país cuando están en la misma habitación? Sus equipos probablemente estén dotados de puertos de infrarrojos o *Bluetooth* que les permitan formar una red para la ocasión. En algunos casos, simplemente no habrá infraestructuras de apoyo. Pensemos en poblaciones aisladas o de orografía difícil, situaciones de emergencia, desastres naturales donde las infraestructuras hayan desparecido, etc.

Otro ejemplo son las denominadas PAN (*Personal Area Networks*) o piconets: redes formadas por los dispositivos de una persona, como su reloj, su agenda y su teléfono móvil. Una red así puede querer entrar en contacto con la red de otra persona que en ese momento esté próxima.

La capacidad de desplegarse inmediatamente y la no dependencia de un único punto de fallo hace a estas redes muy interesantes para el uso militar, de hecho uno de los orígenes de esta idea está en la agencia de proyectos de investigación avanzada para la defensa (DARPA, *Defense Advanced Research Projects Agency*) del ministerio de defensa de Estados Unidos.

El campo militar es posiblemente el más desarrollado actualmente, el ejército estadounidense ya dispone de un sistema basado en este tipo de redes, el FBCB2 (*Force XXI Battle Command, Brigade-and-Below*). Uno de sus objetivos es distinguir las fuerzas propias de las fuerzas del enemigo, ofreciendo a los soldados una visión del campo de batalla similar a la de un videojuego. Los equipos de la generación inmediatamente anterior estaban basados en comunicaciones por satélite, con latencias de cinco minutos. En abril de 2003 el FBCB2 se utilizó en la segunda guerra del golfo, lo que supuso probablemente el primer uso bajo fuego real de una red MANET.

Otro motivo por el que una red MANET puede ser ventajosa es el coste.

Aunque exista una infraestructura de red, si pertenece a una entidad ajena es muy posible que nos cobre por su uso, mientras que si tenemos nuestros equipos desplegados dispondremos ya de una red sin coste adicional. Por ejemplo los coches que pasan por una autopista podrían formar fácilmente una red MANET, independiente de su capacidad de conectarse a otras redes como GSM, o similar. Por último, supongamos que tenemos estaciones capaces de comunicarse empleando un satélite. Estos equipos de comunicaciones son caros, pero bastaría con que algunos tengan capacidad de conectarse al satélite para que todos dispusieran de conectividad. Y no todos los capaces de conectarse al satélite necesitarían estar conectados simultáneamente.

Aunque se puedan pensar muchos usos, la killer ap (programa de ordenador útil para los usuarios, que provoca un aumento importante en las ventas de cierto hardware o sistema operativo necesario para su funcionamiento) puede ser cualquier otra aplicación que hoy no imaginamos. En todo caso, para nosotros y para muchos especialistas resulta evidente que el potencial de este tipo de redes es muy grande.

2.3.6. Tipos

Las redes MANET pueden clasificarse según varios criterios. Aquí nos centraremos en su arquitectura y su aplicación.

Pueden operar de forma autónoma o pueden estar conectadas a otras redes, como Internet. De acuerdo a esto, distinguimos dos tipos de redes:

- **Redes aisladas:** formada únicamente por nodos móviles que se han asociado libremente para constituir la red.

- **Redes subordinadas:** son redes MANET en las que uno o más nodos están conectados a algún tipo de red externa. A los nodos que proporcionan acceso a las redes adyacentes se les denomina *gateways*. Este tipo de redes implican nuevos problemas de diseño frente a las redes aisladas. La interconexión con otras redes condiciona el proceso de asignación de

direcciones, que deben cumplir algunas condiciones para evitar colisiones de direcciones con las redes adyacentes y garantizar su correcto funcionamiento. Además, para dar soporte a este tipo de redes necesitamos proporcionar un mecanismo de detección y mantenimiento de *gateways*.

Hasta ahora hemos tenido en cuenta exclusivamente la estructura de la red. Si embargo, se pueden distinguir distintos tipos de redes según su aplicación y el tipo de nodos que la forman. Basándonos en este criterio cabe distinguir las redes de sensores, cuya investigación y desarrollo está centrando grandes esfuerzos en nuestros días. Vamos a comentar brevemente en qué consisten este tipo de redes.

I. Redes de sensores.

Desde hace algunos años, han comenzado a emerger las WSN (*Wireless Sensor Networks*). Los sensores son fuentes de información tan variados como lo son las medidas que realizan. Los hay de temperatura, de luminosidad, de presión, de humedad, de velocidad, de aceleración, de presencia, de volumen y un sin fin de magnitudes que se nos ocurran. Si a estos sensores que nos reportan información valiosa para nuestras vidas, les añadimos la capacidad de comunicación inalámbrica y la posibilidad de formación de redes MANET, obtenemos las WSN, que están teniendo un auge cada vez mayor debido principalmente a la multitud de aplicaciones que se están desarrollando.

Encontramos estas aplicaciones sin más que mirar a nuestro alrededor: la contaminación de una ciudad se mide con unos sensores de polución, de CO2, Ozono, etc. que, envían los datos en tiempo real y sin necesidad de ningún tipo de interacción por parte humana a un centro de control, y cuando ésta sobrepasa ciertos niveles, se generan unas alarmas para que se tomen las medidas oportunas .En los edificios existen sensores de presencia, que se conectan con el centro de control y emiten alarmas cuando detectan presencias en momentos en que no debiera haber nadie. También esta información procedente de los sensores puede ser procesada y provocar una acción correctora. Así por ejemplo en una bodega, con sensores de humedad

repartidos por toda la planta, cuando se detecta alguna zona en la que la humedad supera cierto umbral se encienden unos ventiladores o se encienden unos microaspersores en caso de defecto de humedad.

En definitiva, las aplicaciones de este tipo de redes son múltiples, desde aplicaciones de seguimiento, de seguridad, de salud, de gestión...

Modos de uso:

Las redes de sensores pueden operar de las siguientes formas:

- Monitorización continua:
  - Nodos midiendo los mismos parámetros en un área de interés. Envío periódico de la información recogida.
  - Aplicación: control de la agricultura, microclimas, etc.
- Monitorización basada en eventos:
  - Nodos monitorizando entornos continuamente. Pero sólo hay envío de información cuando ocurre algún evento.
  - Aplicación: control de edificios inteligentes, detección de incendios, aplicaciones militares, etc.
- Localización y seguimiento:
  - Los nodos se usan para etiquetar y localizar objetos en una zona determinada.
  - Aplicación: rastreo de animales, seguimiento de un trabajador, etc.
- Redes híbridas:
  - Escenarios de aplicación que contienen aspectos de las tres categorías anteriores.

Tipos de redes de sensores.

Existen algunas tecnologías de redes de sensores que se están desarrollando en la actualidad o que, incluso se han convertido ya en estándares.

ZigBee: Comenzaremos por ZigBee, un estándar reciente para la normalización de redes de sensores, promovido por un consorcio de empresas:

la *ZigBee Alliance*. Define un sistema completo de redes inalámbricas con baja velocidad de transferencia de datos para dispositivos muy sencillos, muy baratos y de un consumo tan bajo como para ser capaces de funcionar meses o años sin recargar sus baterías. Para los niveles físico y de enlace, ZigBee confía en el estándar de comunicaciones IEEE 802.15.4, al que añade un nivel de red, de seguridad y un marco de trabajo para las aplicaciones (*application framework*), quedando las aplicaciones y los perfiles de usuario fuera del estándar. En una red 802.15.4 pueden operar dispositivos de funcionalidad completa (FFD, *Full Funcion Device*) y dispositivos de funcionalidad reducida (RFD, *Reduced Funcion Device*). Los FFD pueden comunicarse tanto con FFD como con RFD, mientras que los RFD solo pueden hacerlo con un FFD. El nivel de enlace ofrece topología en estrella, el nivel de red permite la formación de redes MANET con un protocolo basado en AODV. Combinando redes MANET con configuraciones en estrella resulta la topología en árbol. Estas tres topologías (*star, mesh y cluster-tree*) ofrecen mucha más versatilidad que por ejemplo *Bluetooth*, que está limitado una configuración en modo cliente-servidor.

Motas: En 2001 fue acuñado el término polvo inteligente (*smart dust*) para redes de sensores donde cada nodo tendría un volumen de un milímetro cúbico, dimensiones con las que es muy difícil trabajar actualmente. Una plataforma muy extendida hoy, mucho más práctica y acorde con la tecnología actual, son las motas, ligadas a la Universidad de California en Berkeley y su sistema operativo TinyOS. Emplean tecnología ya disponible en el mercado (off the shelf), las más pequeñas son como un reloj de pulsera, las más grandes tienen dimensiones inferiores a las de un paquete de pañuelos de papel. En la figura 1 podemos ver un resumen de las características de las distintas generaciones de motas. Una mota pasa la mayor parte de su vida dormida, se despierta en ciertas ocasiones, hace su trabajo y vuelve a dormir. Esto le permite funcionar entre 300 y 900 días sin reemplazar sus baterías (2 pilas R6 convencionales). La última generación es la plataforma Telos que dispone de un conector USB y usa tramas compatibles con IEEE 802.15.4 (aunque no soporta este protocolo por completo). Las arquitecturas anteriores usaban

componentes de radio no normalizados. En 2006 el precio de una unidad comprada en pequeñas cantidades es de unos 80 dólares.

Nombre	Año	ROM (Kb)	RAM (Kb)	Vel.Trans. (Kbps)	Consumo (mW)
WeC	1998	8	0,5	10	24
René	1999	8	0,5	10	24
René 2	2000	16	1	10	24
Dot	2000	16	1	10	24
Mica	2001	128	4	40	27
Mica2Dot	2002	128	4	38,4	44
Mica 2	2002	128	4	38,4	89
Telos	2004	60	2	250	41

Fig. 2.1: Características de las distintas motas

2.3.7. Ventajas e inconvenientes

Desde finales de los años 90 del siglo pasado la electrónica de consumo ofrece equipos móviles con capacidad de comunicación. Antes de describir las ventajas que ofrecen las redes MANET mostraremos las limitaciones inherentes a las tecnologías centradas en el nivel de enlace, a los protocolos de red convencionales y a la dependencia de infraestructuras previas, y que suponen una serie de inconvenientes a la hora de emplear dichas redes.

Limitaciones a nivel de enlace. En nuestro entorno cotidiano tenemos presentes multitud de dispositivos dotados de comunicaciones inalámbricas. Términos como *WiFi* o *Bluetooth* son algunos de los más repetidos en la publicidad de los grandes almacenes. Pero como veremos a continuación, estas tecnologías, ambas de nivel de enlace, pueden no ser las más adecuadas en ciertas circunstancias, o bien no ser por sí mismas suficientes.

Supongamos ordenadores dotados de un nivel de enlace inalámbrico como por ejemplo *Bluetooth* o el estándar IEEE 802.11, conocido comercialmente como *WiFi*. Actualmente *Bluetooth* permite comunicar varios equipos en un radio de unos 10 metros, 300 en el caso de IEEE 802.11. Si esta distancia resulta ser insuficiente, puede aumentarse de dos formas:

- **Mejora tecnológica en el nivel de enlace:** Es seguro que progresivamente los avances tecnológicos permitirán un aumento del alcance del nivel de

enlace. Pero es un parámetro que crece a un ritmo lento, muy por debajo de lo marcado por la ley de *Moore* (que afirma que la potencia de los ordenadores se dobla cada año y medio, reduciéndose el precio por unidad de potencia a la mitad). El aumento del alcance exclusivamente desde el nivel de enlace tiene dos problemas consustanciales a el:

- Un aumento del consumo de energía, lo cual resulta especialmente serio al ser la batería uno de los elementos que impone más restricciones en cualquier equipo móvil.
- Un aumento del número de equipos compitiendo por el medio, con lo que disminuye el ancho de banda disponible.

- **Uso del nivel de red:** Supongamos un conjunto de estaciones como el de la figura 2. Con una tecnología de nivel de enlace se podrá comunicar A con B. Un protocolo de encaminamiento de nivel de red permitirá comunicar A con D si disponemos de las estaciones intermedias B y C que reenvíen los paquetes de datos.

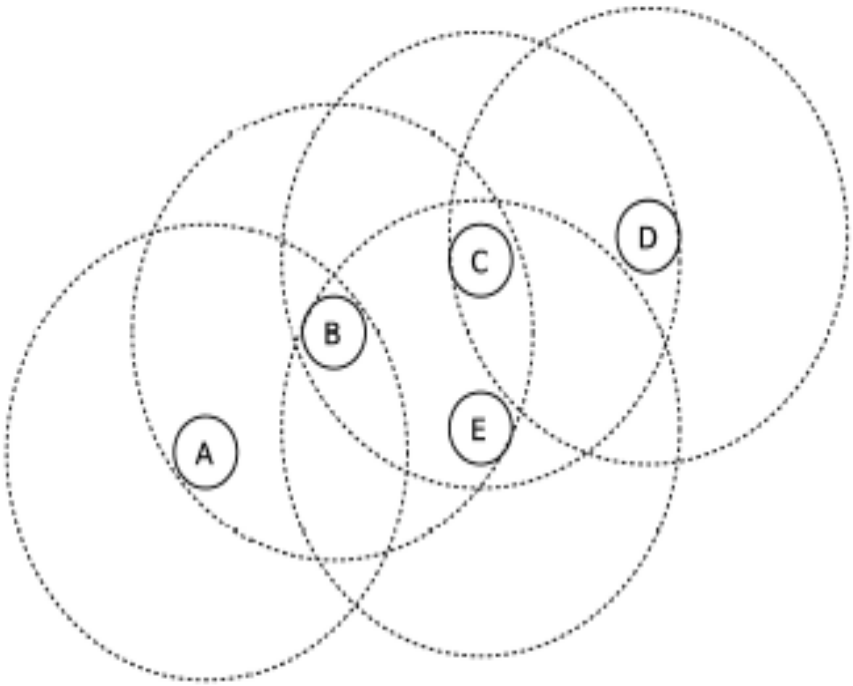


Fig.2.2: Conjunto de estaciones en una red MANET

No importa cuánto mejore el nivel de enlace: añadirle un nivel de red siempre permitirá extender el ámbito de las comunicaciones. El problema que introduce el nivel de red es un aumento de la complejidad del sistema, inabordable para equipos con pocos recursos.

Limitaciones de las infraestructuras:

Volvamos al ejemplo de las redes IEEE 802.11. Sin duda son un gran avance, se emplean típicamente para conectar de forma inalámbrica ordenadores a Internet o a otras redes basadas en el protocolo IP. Un ordenador móvil puede usar enlaces 802.11, pero siempre estará sujeto a permanecer dentro del alcance de un punto de acceso (*Access Point*) preexistente. Si no existe punto de acceso, o si está dañado, desaparece la capacidad de comunicación del dispositivo. Esto mismo es aplicable a telefonía, a UMTS o a cualquier otra red convencional basada en infraestructuras (*infrastructure based networks*), sin la cuales, las estaciones son completamente inútiles a pesar de que por sí mismas serían capaces de enviar y recibir datos.

Limitaciones de los protocolos de red:

Una vez establecida la conveniencia de usar un protocolo de red, en un principio podría plantearse el uso de IP. Las bondades de IP son bien conocidas, es una de las causas del éxito de Internet. Pero eso no significa que el protocolo IP sea el más adecuado en todos los casos. Para dispositivos móviles plantea los siguientes inconvenientes:

- IP es una tecnología de los años setenta, entre otras cosas lleva implícito que una dirección va asociada a un ordenador grande y pesado y por supuesto, fijo. Si un ordenador está quieto, no hay ningún inconveniente en que una dirección sea al tiempo.

- Un identificador, que distingue un equipo de otro

- Un localizador, que aporta la ubicación física donde han de dirigirse los datagramas.

- IP asume esto y encamina a partir de la dirección. Si el ordenador es móvil, ambos conceptos deben separarse. A partir del identificador, es necesaria una entidad que obtenga el localizador en cada momento. Esta entidad se denomina directorio de localización (*Location Directory* o *Mobility Binder*). Este mecanismo adicional introduce complejidad en el sistema, es otra infraestructura que debe existir previamente y es otro posible punto de fallo.

- Los protocolos de encaminamiento convencionales esperan y soportan cambios en los enlaces: averías que se presentan y que se recuperan, nuevos enlaces, etc. Esto es, largas fases de estabilidad interrumpidas por ciertos periodos de cambio. Pero no están diseñados pensando en el estrés permanente provocado por un dispositivo móvil, donde las tablas de encaminamiento convencionales nunca alcanzarían la estabilidad.

- Estos protocolos hacen suposiciones no realistas, como considerar ancho de banda inmutable, o que las baterías tienen una capacidad ilimitada y siempre ofrecen su máxima potencia.

Por tanto, en este escenario los protocolos convencionales o bien tendrán un rendimiento muy pobre, o bien serán simplemente inaplicables. Como alternativa se desarrollan protocolos de encaminamiento para redes MANET. Con frecuencia se les denomina de nivel 2.5, ya que es habitual encontrarlos por encima de protocolos de enlace como IEEE 802.11 y por debajo del protocolo de red IP.

Una MANET de ordenadores supera todas las limitaciones descritas anteriormente. En esta configuración no hay dependencia de unos routers preexistentes: cada equipo es capaz de producir, consumir y encaminar paquetes de datos; cada nodo será capaz de acceder por sus propios medios solo a los nodos más próximos, confiando en la colaboración de sus iguales para llegar a todos los demás. Es un tipo de red que se despliega inmediatamente, donde no hay dependencia de un punto único de fallo. Los nodos típicamente están alimentados por baterías y forman un grafo arbitrario cuya topología puede cambiar rápidamente.

Pueden constituir un sistema autónomo o, adicionalmente, uno o varios de estos nodos pueden estar conectado a otra red como Internet y hacer de pasarela (*gateway*) para los demás. Pero ya no es necesaria cobertura para todos los equipos.

2.3.8. Cuestiones abiertas

Los principales aspectos en los que se investiga actualmente en el ámbito de las redes MANET son:

- **La escalabilidad:** tal vez el principal problema. Hoy con apenas 50 o 100 nodos los resultados empiezan a ser insatisfactorios.

- **El ahorro de energía**: La batería es un bien escaso y muy preciado, es muy importante buscar formas de optimizar su aprovechamiento. Hay técnicas como dormir nodos para despertarlos en determinados intervalos de tiempo, lo que exige una sincronización no trivial. Algunos sistemas permiten transmitir con más o menos fuerza, modificando el alcance de la transmisión.

- **La premisa de la buena fe**: las redes MANET se basan en la cooperación bienintencionada entre estaciones. En las redes ordinarias hay que tratar los problemas ocasionados por nodos maliciosos, pero en estas redes la cuestión es aún más compleja porque no es necesaria la hostilidad para causar daño, basta la ausencia de altruismo. Por ejemplo, si el nivel de la batería está bajo puede ser legítimo no gastarlo retransmitiendo para los demás, o limitarse a los mensajes más urgentes. Pero esto puede dar lugar a abusos.

- **El modelo cliente-servidor** es el habitual en Internet, pero no es adecuado en redes MANET, ya no hay entidades bien conocidas que ofrezcan servicios. Precisamente una pregunta relevante es ¿dónde están los servicios?, lo que resulta difícil en entornos tan cambiantes. Los servicios pueden buscarse al mismo tiempo que las rutas, pero esto va contra el modelo de capas, cuya bondad está demostrada. Una alternativa es

basarse en direcciones *multicast* bien conocidas, esto parece adecuado para servicios básicos como DNS o DHCP, aunque en redes MANET es un tema abierto.

- **La seguridad**: lo que está en el aire puede ser capturado fácilmente. La solución es cifrar las comunicaciones, pero la seguridad y el cifrado se basan en una distribución segura de claves y en una autoridad certificadora centralizada, y esto último es casi una contradicción en términos tratándose de redes MANET. Además, el ancho de banda y la capacidad de procesamiento requerido para hacer seguras las comunicaciones pueden suponer una carga muy importante.

- **El *multicast* (multidifusión):** que puede ahorrar drásticamente ancho de banda en distribución masiva. No está claro si debe incluirse en este nivel de los algoritmos. Por un lado, los problemas de multidifusión pueden ser de naturaleza lo bastante específica como para que no merezca la pena mezclarlos con otras cuestiones. Pero por otra parte, los protocolos dedican mucho esfuerzo a conocer el estado de nodos intermedios que cambian continuamente. Si el *multicast* se convierte en una acumulación de *unicast*, muy posiblemente se estarán desperdiciando recursos.

- **La simetría en los enlaces:** si la estación A puede transmitir a la estación B, lo más habitual es que B pueda transmitir a la estación A. Pero en algunas tecnologías concretas no sucede esto, lo que complica notablemente todos los algoritmos.

- **La gran variedad de medios físicos distintos:** todos incompatibles entre sí. Cuando todos los nodos emplean la misma tecnología en el nivel físico, pueden trabajar juntos, pero solo en ese caso. Esto es un serio inconveniente para llegar a un protocolo estándar.

- **La ubicación en la torre de protocolos**: normalmente se hace en nivel de red. Esto presenta circunstancias favorables. En las tablas de encaminamiento se trabaja con direcciones de red que las aplicaciones resuelven en direcciones de enlace en la dirección de enlace del propio nodo destino si es accesible directamente, o bien la dirección de enlace del

nodo que hace de *router* y que se lo hará llegar. Si la movilidad está en el nivel de enlace, entre otras cosas hay que resolver direcciones de enlace con otras direcciones de enlace, lo que resulta poco natural.

- **La calidad de servicio (QoS)**: podemos indicar si los enlaces son válidos o no, pero apenas hay expresividad para indicar cambios en su calidad.

- **La adaptación de las aplicaciones al ancho de banda disponible:** Usando enlaces inalámbricos, el ancho de banda es típicamente un orden de magnitud inferior al cable. Sería deseable que las aplicaciones fueran conscientes de ello, y en cada caso adapten la información enviada, tal vez reduciendo el número de píxeles por segundo si se trata de vídeo, el número de colores o la resolución en las imágenes, u omitiendo en páginas Web animaciones de utilidad dudosa.

- **Las máquinas de recursos limitados**: este problema obliga a que nuestros mecanismos sean ligeros, para poder ejecutarse en máquinas con una capacidad computacional limitada, no necesiten almacenar mucha información debido a la escasez de memoria, y no consuman demasiada energía para aumentar la duración de las baterías. El tema de la eficiencia energética de las comunicaciones en redes MANET ha dado pie a numerosas investigaciones y publicaciones.

3. Autoconfiguración de direcciones IP en redes MANET

Los nodos de una red necesitan de algún mecanismo para intercambiarse mensajes. El protocolo TCP/IP permite comunicar a los diferentes nodos de una red asociando a cada nodo de la misma una dirección IP distinta.

En redes cableadas o en redes inalámbricas con infraestructura se dispone de un servidor o de un nodo que actúa como tal que asigna correctamente las direcciones IP.

En redes MANET no se dispone de una entidad centralizada que pueda realizar esta función. Por tanto, es necesario un protocolo que realice la configuración de la red de forma dinámica y automática, que utilizará todos los nodos de la red (o sólo una parte de ellos) como si fuesen servidores que gestionan direcciones IP.

3.1. Problemas de la autoconfiguración en las redes MANET

Debido a la tolopología dinámica de las redes MANET, con el constante movimiento de nodos que pueden salir y entrar a la red frecuentemente e incluso simultáneamente, los protocolos de autoconfiguración se enfrentan a diversos problemas para garantizar la unicidad de las direcciones IP y permitir el particionado o la unión de redes.

Para garantizar el correcto funcionamiento de la red los protocolos de pretenden lograr los siguientes objetivos:

- **Lograr la unicidad de las direcciones IP:** Asegurar que dos o más nodos no obtengan la misma dirección IP.

- **Funcionar correctamente:** Una dirección IP está asociada a un nodo solamente por el tiempo que permanece en la red. Cuando un nodo deja la red, su dirección IP debe quedarse disponible para ser asociada a otro nodo.

- **Solucionar los problemas derivados de la pérdida de mensajes:** En caso de que algún nodo falle u ocurra pérdida de mensajes, el protocolo debe actuar lo suficientemente rápido para evitar que dos o más nodos posean la misma dirección IP.

- **Permitir el encaminamiento multi-salto** (*multi-hop*)**:** Un nodo no se configurará con una dirección IP si no hay ninguna disponible en toda la red. De esta forma, si cualquier nodo de la red posee una dirección IP libre ésta debe asociarse al nodo que está solicitando una dirección IP, aunque esté a dos o más saltos de distancia.

- **Minimizar el tráfico de paquetes adicionales en la red:** El protocolo debe minimizar el número de paquetes intercambiados entre los nodos en el proceso de autoconfiguración. En otras palabras, el tráfico de paquetes de control debe perjudicar lo menos posible al tráfico de paquete de datos ya que, en caso contrario, el rendimiento de la red disminuiría.

- **Verificar la existencia de solicitudes concurrentes de dirección IP:** Cuando dos nodos solicitan una dirección IP en el mismo instante de tiempo, el protocolo debe realizar el tratamiento pertinente para que no sea suministrada la misma dirección IP a los dos nodos.

- **Ser flexible al particionamiento y a la fusión de redes MANET:** El protocolo debe poder lograr la fusión de dos redes MANET distintas así como el particionamiento en dos o más redes.

- **Realizar la sincronización:** El protocolo debe adaptarse a los rápidos cambios de la topología de las redes inalámbricas debido a la frecuente movilidad de los nodos. La sincronización se realiza periódicamente para mantener una configuración lo más actualizada posible de la topología de la red.

3.2. Aplicabilidad de las soluciones estándar

Como se describe en [3], la aplicabilidad de los protocolos estándar es insuficiente para redes MANET. A continuación se presentan dos de estos protocolos.

3.2.1. SLAAC/NDP

SLAAC (*StateLess Address AutoConfiguration* [4]), que data de 1998, es un estándar que permite la autoconfiguración automática de una dirección IPv6 sin necesidad de un nodo encaminador (*router*). Para ello se ayuda del protocolo NDP (*Neighbor Discovery Protocol* [3]), un estándar para transmitir los mensajes y descubrir a sus vecinos.

Un nodo crea automáticamente una dirección IPv6 uniendo su identificador de host (suele ser la dirección MAC) con un prefijo local conocido y realiza un proceso DAD (*Duplicate Address Detection*) mediante la difusión (*broadcast*) de mensajes NDP a los vecinos.

Si la dirección IPv6 no es única el proceso de autoconfiguración se detiene y se tendrá que hacer manualmente. Si por el contrario la dirección es única deberá pedir mediante mensajes NDP el prefijo de red y posteriormente volverá a comprobar con DAD si su dirección IPv6 es realmente única.

La aplicabilidad de este protocolo en redes MANET queda limitada porque usa el protocolo NDP para enviar los mensajes y NDP asume que en la red todos los nodos están conectados entre sí. Consecuentemente, sólo soporta un único salto y lo más frecuente es que la red MANET sea multi-salto, no alcanzando a la mayoría de los nodos para realizar los procesos DAD y no pudiendo, por tanto, asegurarse de que la dirección IPv6 obtenida sea única.

3.2.2. DHCP-PD

DHCP-PD [6] (*Dynamic Host Configuration Protocol – Prefix Delegation*) es una opción de DHCPv6 [7] que provee un mecanismo para la delegación de los

prefijos de direcciones IPv6 y permite la asignación automática de uno de estos. Para ello, un nodo que desea obtener una dirección IPv6, manda un mensaje DHCP con la opción *prefix delegation* activada para obtener un prefijo de un servidor DHCP de la red.

La aplicabilidad en redes MANET es insuficiente porque está basado en DHCP, por lo que asume que todos los nodos pueden conectarse ya sea directamente o a través de varios saltos con un servidor DHCP, y debido a la topología de las redes MANET, la conexión directa al servidor DHCP no suele ser frecuente con la consecuencia de que la conexión mediante varios saltos puede producir que el servidor sea inalcanzable.

3.3. Protocolos de autoconfiguración para redes MANET

Los protocolos de autoconfiguración pueden clasificarse dependiendo del modo de gestión de direcciones en *stateless*, *stateful* o *hybrid.*

- ***Stateful*:** Los nodos conocen el estado de la red, es decir, mantienen tablas con las direcciones IP de los nodos.
- ***Stateless*:** La dirección IP de un nodo está gestionada por el mismo, generalmente crean una dirección aleatoria y realizan un proceso de detección de direcciones duplicadas para verificar su unicidad.
- ***Hybrid:*** Combinan mecanismos de los dos anteriores para mejorar la escalabilidad y la fiabilidad de la autoconfiguración. El precio es un alto nivel de complejidad en los algoritmos.

3.3.1. Stateful

I. MANETConf

MANETConf [8] está basado en la existencia de una tabla común distribuida con la que todos los nodos son capaces de asignar direcciones IP manteniéndola actualizada.

Cuando un nodo quiere entrar en la red envía mensajes de difusión y al primero que le conteste lo elige como nodo iniciador y le pide una dirección IP. El nodo iniciador elige una de las direcciones IP libres que hay en la red y antes de asignarla pide permiso al resto de nodos, porque puede darse el caso de que otro nodo la haya elegido para otro o porque puede que las tablas no estén totalmente sincronizadas debido al retardo de los mensajes.

Si la respuesta de los nodos es positiva, le asigna la dirección IP al nodo entrante y lo comunica por difusión para que el resto de nodos actualicen sus tablas.

Si hay algún nodo que no responde, se pone en contacto con él directamente (*unicast*) para obtener respuesta, si aun así no logra obtenerla, dará por hecho que el nodo ha dejado la red, y lo comunicará al resto de nodos de la red para que actualicen sus tablas.

II. EMAP

EMAP (*Extensible Manet Autoconfiguration Protocol,* 2005 [9]) es un protocolo de autoconfiguración que se basa en la idea del protocolo de mensajes REQUEST/REPLAY para su funcionamiento.

La principal ventaja de este protocolo es la posibilidad de hacerlo extensible, es decir, que en el futuro se pueden incluir nuevas funcionalidades que quedan tratadas de forma teórica como el descubrimiento de servidores DNS (*Domain Name Server*).

Este protocolo trata también la posibilidad de comunicaciones exteriores a la red móviles ad hoc a través de Internet.

El mecanismo de descubrimiento de rutas entre nodos sigue la línea del protocolo AODV (*Ad Hoc On-Demand Distance Vector* [10]), explicado en el apartado de protocolos de encaminamiento.

La principal idea de este protocolo de autoconfiguración es la de poseer diferentes direcciones por parte de un nodo no configurado que va a unirse a la red ya creada. De esta manera existen tres direcciones para las comunicaciones interiores, la temporary address, la tentative address, y la movile ad hoc network local address.

Cuando un nodo quiere entrar en la red genera aleatoriamente dos direcciones IP válidas de la red (con dirección de red conocida), y las considera como temporary address y tentative address-. Estas direcciones IP se encapsulan en el mensaje DAD_REP (*Detection Adress Detection REsPonse*) para saber si es una dirección válida. El nodo queda esperando un DAD_REQ (*Detection Adress Detection REQuest*), si transcurre el tiempo de espera para este mensaje el nodo asume que puede usar su tentative address como única, y se la asigna a su interfaz de red. Si este nodo recibe un mensaje DAD_REP a su temporary address, y este mensaje contiene el origen con la tentative address que había propuesto, sabe que esta tentative address está siendo usada y comienza a recrear el proceso creando otro par de direcciones.

En cuanto a la configuración global, tanto para el descubrimiento de puertas de enlace para Internet o para servidores DNS se basa en un sistema de inundación periódica con mensajes GC_REP (*Gateway Control REsPonse*) y GC_REQ (*Gateway Control REQuest*) para puertas de enlace y mensajes DS_REP (*DNS Server REsPonse*) y DS_REQ (*DNS Server REQuest*) para servidores DNS. Estos sistemas siguen con la idea del sistema de detección de direcciones duplicadas descrito en el anterior párrafo.

III. IP Address Assignment in a mobile Ad Hoc network

IP Address Assignment in a mobile Ad Hoc network [9](2002) trata de solucionar el problema de la asignación de direcciones IP mediante la división binaria de bloques de direcciones libres.

Estas divisiones se hacen en potencias de 2, de esta forma, pueden existir nodos con varios o ningún bloque. Además, cada nodo de la red dispone de una tabla con el estado de todos los nodos de la red, es decir, conoce los bloques de direcciones libres de los otros nodos, así como la dirección IP de la interfaz de red de cada nodo.

El proceso de asignación de direcciones puede proceder de cualquier nodo. Siguiendo esta idea, un nodo no configurado (nodo cliente) envía un mensaje de difusión del tipo REQUEST para que un nodo de la red lo configure; al ser un mensaje de difusión podrían producirse varias respuestas, pero elegirá al nodo que realice la primera respuesta (*REPLAY*), y este actuará como nodo servidor. Si este nodo dispone de varios bloques, le entregará uno al nodo cliente, y elegirá la primera IP del bloque como propia. Si por el contrario dispone de un bloque, entonces lo dividirá en 2 partes iguales y le entregará una mitad al nodo cliente, y la otra se la quedará como propia.

Si se diera el caso en que no dispusiera de ningún bloque se proponen ciertas soluciones basadas en la idea de que el servidor busque en sus vecinos próximos si alguno tiene algún bloque disponible. En caso de no encontrar direcciones libres intentará obtener un bloque en vecinos de un salto mayor, y así sucesivamente. Otra solución es buscar el nodo que tenga mayor rango de direcciones libres, para entregar la mitad de este bloque al nodo que está entrando en la red.

La entrada y salida de los nodos puede ser de forma abrupta o voluntaria, para cada caso hay un esquema a seguir. Si la salida es abrupta, el nodo que actuó como servidor en su configuración verá en su tabla de rutas que el bloque del nodo que abandonó la red no está, añadirá el bloque a su bloque de direcciones libres. En el caso de una salida voluntaria el nodo que sale, notifica a algún vecino suyo la intención de marcharse y el vecino busca al nodo que configuró al cliente para que se quede con su bloque.

La mayor ventaja de este protocolo es que funciona bien para unión y división de subredes, ya que soluciona el problema de las direcciones duplicadas que se producen en estos casos. Cada nodo que inicia la red

genera un número aleatorio llamado *PartitionID* que será un número identificativo d la red. De esta manera cuando se produce una división o partición de la red, el primer nodo que se retire de la red original creará otro *PartitioID*. En el momento en que dos redes con *PartitionID* diferente se unan, primero se comprobará la consistencia de sus direcciones IP. Este proceso consiste en comprobar la existencia de dos nodos con una misma dirección. En caso afirmativo se produce un cambio del nodo perteneciente a la red con menor rango de direcciones IP libres. La nueva dirección IP será perteneciente al mayor rango de la red con mayor número de direcciones libres.

3.3.2. Stateless

I. DAD (*Duplicate address detection*): SDAD, WDAD y PDAD

DAD es un proceso que utilizan los protocolos para comprobar la unicidad de las direcciones IP. Este proceso requiere un tiempo relativamente largo para completarse, por lo que se han implementado diferentes soluciones que lo reducen.

Existen 3 tipos de procesos DAD:

- **SDAD:**

  Strong Duplicate Address Detection [12] es la base de los protocolos *stateless*, consiste en un sencillo mecanismo en el que el nodo elige dos direcciónes IP, una temporal y una tentativa.

  La dirección temporal sólo la usara para la inicialización mientras detecta si la tentativa es única o no. El método de detección consiste en mandar un mensaje ICMP destinado directamente a esa dirección, si recibe respuesta, esa dirección IP está usada por lo que reanudará el proceso, si no recibe respuesta, enviará el mensaje un número determinado de veces para asegurar que es única.

  Al ser un mecanismo muy sencillo, no asegura la unicidad de la dirección IP ya que el proceso se limita solo a la fase de inicialización, y para

desconexiones temporales o pérdida de la red no funcionaría. Además, cuando la red es grande y quedan pocas direcciones IP libres, añade mucha sobrecarga hasta que encuentra una dirección IP única.

- **WDAD:**

  Weak Duplicate Address Detection [13] establece la idea de tolerar durante un tiempo las direcciones duplicadas en la red.

  Para ello, cada nodo al iniciarse creará una clave que enviará siempre junto a su dirección IP. Cuando un nodo reciba un mensaje comprobará en su tabla si esa dirección IP ya está asignada y mirará si las claves coinciden, si no coinciden, marcará esa dirección como inválida y se tomarán acciones para que sean únicas (estas acciones no están definidas en WDAD).

  Este proceso tiene que soportar la identificación de un nodo mediante un par clave-IP y depende totalmente del protocolo de encaminamiento, solo funcionará con un proactivo que actualiza las rutas constantemente, pero con un reactivo habrá nodos que nunca puedan detectar la duplicidad de direcciones IP.

  No añade sobrecarga adicional al protocolo de encaminamiento, pero en cambio si añade la sobrecarga de enviar siempre junto a la dirección IP la clave.

- **PDAD:**

  En *Passive Duplicate Address Detection* [14] la idea se basa en que en vez de detectar o resolver direcciones IP duplicadas enviando información de control, cada nodo investiga y deduce si existe una dirección duplicada por eventos que nunca ocurrirían si todas las direcciones IP fueran únicas.

Se proponen tres detecciones pasivas, que son necesarias unir para el correcto funcionamiento de la detección:

  - **PDAD-SN (*Sequence Numbers*).** Este sistema se basa en la idea de que los protocolos de encaminamiento usan números de secuencia en

sus mensajes para actualizar las rutas. Usando estos números de secuencia, y la idea de que dos nodos con una distancia entre ellos de dos saltos no tienen el mismo vecindario, se solucionan algunos conflictos. Además, se tiene en consideración la posibilidad de que estos números de secuencia lleguen al máximo y empiecen a producirse números desde cero de nuevo.

- **PDAD-LP (*Locality Principle*).** De menor potencia que el anterior, se basa en la frecuencia de actualización de las tablas de rutas. En función de esta frecuencia se pueden detectar direcciones duplicadas tomando un umbral de tiempo para visualizar el estado de las tablas de rutas. Hay que tener en cuenta el protocolo de encaminamiento usado, debemos considerar diferentes umbrales, ya que se puede dar el caso de que dos mensajes con el mismo origen se confundan con una dirección duplicada, si el tiempo es demasiado corto, y por consiguiente el protocolo modifica las rutas demasiado rápido.

- **PDAD-NH (*Neighborhood*).** Teniendo en cuenta que un nodo conoce sus vecinos, y los de un nodo que haya enviado un paquete del estado de su enlace, diferencia si hay conflicto o no en función de si en un paquete, el origen de este mensaje es un vecino, y contiene la dirección.

La ventaja es que no añade sobrecarga a la red adicional, pero solamente se puede utilizar con protocolos de encaminamiento proactivos.

II. APAC

APAC (*Agent Based Passive Autoconfiguration for Large Scale manets,* 2007 [15]) es un protocolo de autoconfiguración basado en PDAD. Su característica principal es el uso de ciertos nodos que centralizan el reparto de direcciones.

El mecanismo por el que un nodo configura su dirección IP al entrar en la red consiste en preguntar si tiene a un salto de distancia algún nodo de tipo AA (*Address Agent*). En ese caso, el nodo AA le proporcionará una dirección IP.

En caso de no tener respuesta de ningún nodo AA, el nodo entrante se configura para funcionar en modo AA, y hará de servidor de direcciones para los próximos nodos que lleguen. Cuando se configura como AA, el nodo genera aleatoriamente un número identificador *agentID*, para poder formar una tabla con las direcciones que asignará a los nodos que lleguen. Esas direcciones son de la forma *agentID*+*hostID*.

Cuando un nodo se mueve en la red y sale del radio de cobertura del AA que le proporcionó su dirección IP, deberá pedir otra dirección a otro nodo AA que tenga dentro de su nuevo radio de cobertura. Esto se complementa con un mecanismo para no interrumpir las comunicaciones en curso.

Al darse esta situación, el AA anterior marcará su dirección IP como libre para poder ser asignada a algún otro nodo más adelante.

La detección de direcciones duplicadas se realiza mediante el proceso PDAD. Una vez detectado algún conflicto, se informa al AA que asignó esa dirección IP conflictiva. Ese nodo AA entonces generará un nuevo *agentID* y avisará a todos los nodos dependientes de él para que cambien su dirección del tipo *agentID*+*hostID* al nuevo *agentID*.

En cuanto a la división y unión de redes, se utiliza el mismo mecanismo explicado anteriormente. En caso de división, los nodos AA marcarán las direcciones que hayan salido de la red como libres. Y en el caso de unión de dos redes, el mecanismo para la detección de direcciones duplicadas sigue funcionando correctamente.

III. AROD

En AROD (*Address autoconfiguration with Address Reservation and Optimistic duplicated address Detection for mobile ad hoc networks* [16]), la reserva de direcciones se basa en la existencia de unos nodos que tienen una dirección IP reservada para entregársela a los nodos que entren nuevos.

Existirán dos tipos de nodos:

- Agentes tipo 1 con una dirección IP reservada, aparte de las direcciones IP que tengan sus interfaces de red. Cuando un nodo entra en la red, se le asignará esta IP reservada inmediatamente.

- Agentes tipo 2, que no tienen direcciones IP reservadas. Si un nodo que entra nuevo le pide una dirección IP a uno de estos, pide prestada la dirección reservada de uno de sus vecinos que sea de tipo 1, y se la asigna al nuevo inmediatamente.

Establece un mecanismo para que la red no se quede sin nodos de tipo 1, cada vez que se asigna una dirección IP a un nuevo nodo, el nodo que ha entregado la dirección genera dos direcciones IP aleatorias (una para sí mismo y otra para el nuevo nodo que ha entrado) y se hace un proceso DAD para detectar si esas direcciones son únicas.

Una vez realizado el proceso se pueden dar las siguientes posibilidades:

- Las dos direcciones IP son únicas, por lo tanto, los dos nodos se convierten en nodo tipo 1.

- Si solo una es única, se convertirá en nodo tipo 1 el que dio la dirección IP.

- Si ninguna es única, los dos nodos se quedarán como tipo 2.

Este protocolo considera su proceso de detección de direcciones duplicadas como un proceso DAD optimista, porque solamente se lleva a cabo una vez cuando entra un nodo y se le asigna una dirección IP reservada.

Contempla la posibilidad de cambiar el número de direcciones IP reservadas para los nodos tipo 1, si aumenta el número, la latencia de asignación de direcciones IP es menor, pero por el contrario aumenta la sobrecarga al tener que realizar más procesos DAD, y viceversa. Esta posibilidad permite variar latencia vs sobrecarga.

3.3.3. Hybrid

I. HCQA

HCQA (*Hybrid Centralized Query-based Autoconfiguration)* [17] fue el primer protocolo de autoconfiguración híbrido.

Un nodo que quiera entrar en la red realiza el proceso SDAD tal y como está explicado en el apartado 3.3.2.1. Si el proceso es exitoso, el nodo deberá registrar su dirección IP tentativa con una AA (*Address Authority*). Para ello, esperará un mensaje del AA y cuando reciba ese mensaje enviará una petición de registro y el AA se lo confirmará. El nodo al comenzar todo este proceso inicia un contador, si ese contador expira, volverá a empezar de nuevo el proceso hasta poder registrar la dirección IP.

Cuando se crea la red, el primer nodo se convierte en AA, elige un identificador para la red único (por ejemplo la dirección MAC) y mediante mensajes de difusión la anuncia periódicamente para identificar la red. Si algún nodo no la recibe se considera que la red ha sido particionada y creará su propia red convirtiéndose en AA.

Este protocolo añade robustez al proceso SDAD y garantiza la no duplicidad de las direcciones IP y a la vez proporciona un buen mecanismo de partición de redes.

Pero tiene dos principales problemas, primero la sobrecarga producida por el proceso SDAD y los mensajes periódicos del AA, y segundo, que la red depende de una entidad central con la que todos los nodos deben comunicarse directamente para poder registrar su dirección IP, con lo que se añade mucha latencia en la entrada de los nodos a la red.

II. PACMAN

PACMAN (*Passive Autoconfiguration for Mobile ad hoc Networks, 2004* [18]), es un protocolo de autoconfiguración pasivo para redes MANET.

Utiliza elementos de protocolos con y sin estado, por lo que podría considerarse híbrido en cierto modo.

Su funcionamiento se basa en que cada nodo se asigna a sí mismo una dirección al entrar a la red, y en la monitorización pasiva de las comunicaciones para la detección de direcciones duplicadas.

Para conseguir la mínima sobrecarga en las comunicaciones, se comparte información entre las diferentes capas de red. En concreto se monitoriza la información que maneja el protocolo de encaminamiento.

El método usado para elegir la propia dirección IP que se usará sigue un algoritmo probabilístico. Para intentar que la probabilidad de elegir una dirección IP actualmente en uso por otro nodo sea cercana a cero, este algoritmo tiene en cuenta, entre otros factores, una tabla de asignaciones. Esta tabla se crea con información extraída del protocolo de encaminamiento sobre las direcciones IP que ya están en uso.

PACMAN usa el proceso PDAD para monitorizar las comunicaciones en busca de direcciones duplicadas. Esto es necesario debido a que el mecanismo utilizado para la asignación de direcciones no garantiza unicidad (aunque intenta reducir la probabilidad de colisión), y a que pueden producirse unión de redes que contengan nodos con las mismas direcciones IP.

A grandes rasgos hay dos tipos de eventos que indican duplicidad de direcciones IP:

Por un lado están los eventos que nunca ocurren si la dirección es única, y siempre si la dirección está duplicada. Estos eventos confirman que hay un problema detectado.

Y por otra parte tenemos los eventos que ocurren pocas veces si la dirección es única, y a menudo si la dirección está duplicada. De este modo se detecta la posibilidad de que existan problemas, por lo que se trata de algoritmos probabilísticos.

Cuando se detecta que dos nodos están usando la misma dirección IP, se le notifica el problema a uno de ellos mediante un mensaje unicast para que cambie su dirección.

Además, se tiene en cuenta el problema que surge cuando se cambia una dirección que tiene alguna comunicación en marcha. Para solucionarlo, al cambiar de dirección un nodo avisa a los nodos con los que tiene comunicaciones en curso de su nueva dirección IP, para que hagan un encapsulamiento de los mensajes adecuadamente.

3.4. Resumen

Las redes MANET necesitan de un proceso de configuración dinámico y automático debido a los diversos problemas que conllevan sus características. Los protocolos estándar utilizados en otro tipo de redes son insuficientes para las redes MANET por lo que se necesita implementar nuevos protocolos de autoconfiguración que pretendan cumplir los requisitos que una red MANET necesita.

Existen varios protocolos en la actualidad, pudiéndolos diferenciar según la información de estado que guarden los nodos, pero ninguno llega a ser definitivo para este tipo de redes debido a la alta complejidad de estas.

Por ello, es necesario crear nuevas ideas en nuevos protocolos para poder conseguir un protocolo que sea capaz de cumplir en su totalidad todas las funciones que una red MANET requiere.

4. Encaminamiento en redes MANET

El concepto de encaminamiento engloba dos actividades, determinar los caminos óptimos entre dos nodos y transferir los paquetes de información a través de la red.

Estos protocolos calculan los caminos óptimos a partir de unas métricas, diferentes entre unos u otros, pudiendo ser el número de saltos, la distancia, el consumo, etc.

El carácter dinámico de las redes MANET obliga a hacer que los algoritmos cambien sus rutas debido al movimiento de los nodos o a la aparición o desaparición repentina de estos. Cada nodo de una red MANET puede actuar también como router cuando el camino entre dos nodos necesite de más de un salto.

El protocolo de encaminamiento necesita proveer una forma de establecer y mantener las rutas a pesar de posibles caminos rotos donde tendrán que encontrar una ruta alternativa.

El objetivo principal de los algoritmos de encaminamiento es el establecimiento correcto y eficiente de la ruta entre un par de nodos de manera que los paquetes sean entregados de manera confiable.

Los protocolos de encaminamiento para redes MANET deben satisfacer básicamente los siguientes criterios [19]:

- **Señalización mínima:** La reducción de los mensajes de control ayuda a conservar la capacidad de batería y la comunicación de los nodos.
- **Tiempo de procesamiento mínimo:** Se requieren algoritmos con cálculos computacionales que no sean excesivamente complejos para disminuir el tiempo de procesamiento y alargar de esta forma el tiempo de vida de la batería.

- **Mantenimiento en condiciones de topología dinámica:** El algoritmo deberá ser capaz de localizar una nueva ruta rápidamente cuando se rompe un enlace.

- **Modo de operación distribuido:** Propiedad esencial de las redes MANET.

- **Libre de bucles:** Se pretende evitar el problema de tener paquetes circulando perdidos por la red.

- **Modo de operación '*sleep*' o inactivo:** Los protocolos de encaminamiento deberán estar preparados para afrontar aquellos periodos de tiempo en los cuales los nodos frenan su actividad y permanecen inactivos para ahorrar energía.

- **Soporte de enlaces unidireccionales:** Los protocolos de encaminamiento en muchas ocasiones han sido diseñados y funcionan correctamente sólo con enlaces bidireccionales y esto no debería ser así, porque en la práctica podemos encontrarnos con la existencia de enlaces unidireccionales que sean clave para el intercambio de información en redes MANET.

Se han diseñado numerosos protocolos de encaminamiento para redes MANET atendiendo a estos criterios. A continuación se realiza una posible clasificación de los protocolos de encaminamiento con el fin de dividirlos en una serie de categorías y se introducen ejemplos en cada una.

4.1. Clasificación de los protocolos de encaminamiento

Desde que se crearon los primeros protocolos de encaminamiento se clasificaron de diversas formas, las resumimos a continuación [20].

- **Uniformidad:** Existen los uniformes y los no uniformes. En los primeros todos los nodos realizan el mismo rol que los demás, no hay distinción entre ellos. En los no uniformes, por el contrario, hay una diferencia entre los

nodos, disminuyendo el número de nodos que participan en las tareas de encaminamiento.

- **Información de estado:** Dependiendo de que información guarden los nodos diferenciamos entre protocolos basados en la topología y basados en el destino. Los que se basan en la topología de la red, poseen una completa información sobre la topología, y los que se basan en destino, manejan vectores distancia donde cada nodo intercambia información de las rutas con sus vecinos.

- **Forma de detectar las rutas:** Es sin duda la clasificación más reconocida y más usada, clasificándolos en proactivos, reactivos e híbridos.

    - **Proactivos:** Cada nodo guarda información de cómo llegar a cualquier nodo de la red en una tabla de rutas y a su vez intercambia esta información con sus vecinos. Esta información se refresca periódicamente por si han ocurrido cambios en la red. La principal ventaja es la latencia, que es mínima ya que las rutas a cualquier nodo están siempre disponibles y actualizadas. Como desventajas tiene el consumo de energía que provoca los refrescos de la red y las sobrecargas de mensajes que se producen cuando se actualizan periódicamente las rutas.

    - **Reactivos:** Estos protocolos intentan evitar la sobrecarga que producen los proactivos, para ello eliminan la actualización periódica de las rutas, y calculan las rutas exclusivamente cuando se necesitan. El proceso para calcular las rutas se basa en la inundación por toda la red. Al contrario que los proactivos, evitan la sobrecarga, pero producen una gran latencia, ya que cada vez que se quiera enviar un paquete, se necesitará descubrir la ruta.

    - **Híbridos**: Es una combinación de los reactivos y los proactivos, intentan minimizar los problemas que estos dos conllevan. Su idea se basa en

que los nodos más cercanos funcionen entre si proactivamente y los lejanos reactivamente. La parte reactiva controla la sobrecarga y la proactiva la latencia, como solo los nodos más cercanos funcionan proactivamente, se evita hacer rutas a nodos lejanos, que posiblemente nunca se vayan a utilizar.

En la figura 4.1 se puede observar un esquema de clasificación de los protocolos.

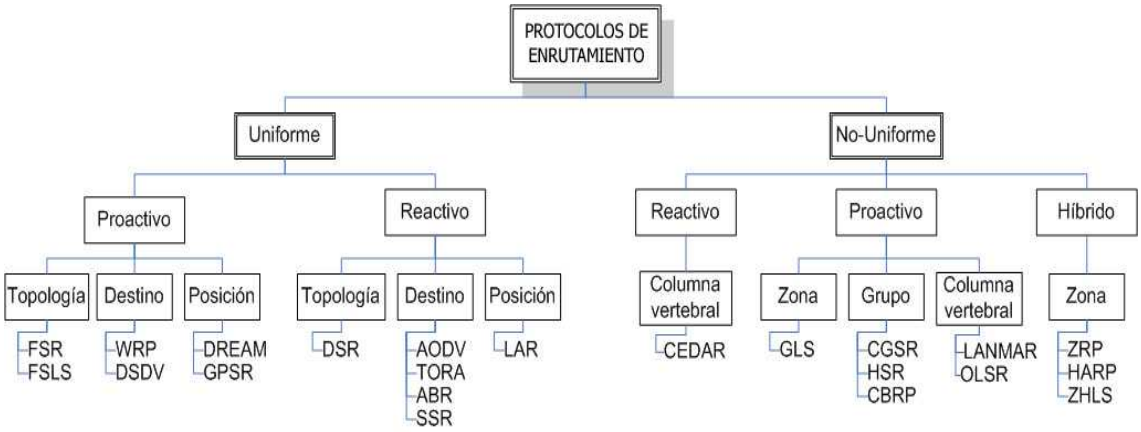


Fig. 4.1: Posible clasificación de los protocolos en redes MANET

4.2. Protocolos reactivos

4.2.1. AODV

Creado por Charles E. Perkins (*Ad Hoc On-Demand Distance Vector* [8]) como evolución de su anterior protocolo DSDV (*Destination-Sequenced Distance-Vector Routing* [21]). Mantuvo la idea de mantener números de secuencia y tablas de encaminamiento pero añadió el concepto de encaminamiento bajo demanda (protocolo reactivo), es decir, solo se guarda información de los nodos que intervengan en la transmisión de datos. La optimización primordial que se consiguió en relación a su anterior diseño fue el decremento del tiempo de proceso, disminución del gasto de memoria y reducción del tráfico de control por la red. Además AODV es muy cuidadoso con las rutas, manteniéndolas en caché mientras son necesarias e inhabilitándolas cuando su información no es útil.

El encaminamiento se produce salto a salto, mediante un vector de distancias, este protocolo se caracteriza de forma global en que:

- Nadie tiene un grafo completo del estado de la red.
- Para cada posible destino sólo se conocen el primer salto por donde debe encaminarse y la distancia a la que se encuentra.

Para distinguir la información moderna de la antigua, se emplean horas lógicas (identificador de nodo, no de secuencia) esto se usa de forma local, es decir, no hay un reloj absoluto. Toda la información que se incluye en la tabla de estado de enlaces, o tabla de encaminamiento, lleva la hora lógica de quien la generó, ya que es posible que cierta información recién recibida sea más antigua que otra recibida anteriormente.

Este mecanismo se realiza debido a que toda información acaba caducando a los pocos segundos.

Funcionamiento General:

Cuando se necesita una ruta desde un origen hasta un destino, se inunda la red con peticiones RREQ (*Route REQuest*). Cuando un RREQ llega al destino buscado, o a alguien que conoce una ruta para el destino, se genera una respuesta RREP (*Route REsPonse*). Este RREP sabe volver al origen porque la inundación del mensaje RREQ fue creando el camino de vuelta. Cuando el RREP va volviendo al origen, va creando el camino de ida. Una vez que el origen ha recibido el RREP, ya puede enviar datagramas, que seguirán el camino de ida generado por el envío de mensajes anteriores.

Para la formación del camino de vuelta, cuando las peticiones de ruta inundan la red, las tablas apuntan hacia el origen. Si un nodo no sabe responder la petición de ruta, la reenvía. Las tablas almacenan, en cada entrada, la hora lógica en el origen. La petición de ruta viaja en sentido inverso.

En cuanto a la formación del camino de ida, el RREQ tiene éxito si llega al destino o a alguien que sabe llegar al destino. Cuando el RREP vuelve, las tablas de los nodos apuntan hacia el destino. También se conserva la hora

lógica del destino. Si transcurrido cierto tiempo no llega un RREP, se borra la información.

Cada petición RREQ lleva encapsulado:

- Origen de la petición y destino buscado.
- Identificador de petición (para controlar inundación).
- Hora lógica en que el destino envió la última ruta conocida (cero si no se conocía ninguna). Esto garantiza que nadie responda una ruta más vieja de la ya conocida.
- Hora lógica en que el origen envió la petición (Si el origen se mueve y envía nuevas RREQ, la información sobre el nuevo camino de vuelta debe reemplazar a la información antigua).
- TTL

Cada respuesta RREP (viajando desde el destino hacia origen) incluye:

- Origen de la petición y destino buscado
- Hora lógica del destino
- Distancia del origen al destino

Para el mantenimiento de rutas se produce el siguiente mecanismo. La detección de un movimiento se percibe como un cambio en el vecindario, además cada nodo mantiene información sobre sus vecinos con mensajes *HELLO* periódicos, de *TTL*=1.

Cada nodo mantiene información sobre rutas vivas (activas), por este motivo si un cambio en el vecindario afecta a una ruta viva, se notifica un error para generar nuevas búsquedas. Por otro lado, la desaparición de un vecino que no participa en ninguna ruta viva no provoca ninguna acción.

4.2.2. DYMO

DYMO (*Dynamic MANET On-demand* [22]) es un protocolo en el que el descubrimiento de nuevas rutas se realiza mediante paquetes de control especiales.

Cuando es necesario descubrir nuevas rutas, el nodo origen manda mediante difusión un paquete especial llamado *Route Request* (RREQ), que es una petición de ruta hacia el nodo destino. Este mensaje se retransmite de nodo a nodo, y cada uno de estos nodos intermedios que reciben y reenvían el mensaje lo modifican anotándose a ellos como parte de la ruta hacia el nodo destino.

Cuando este mensaje RREQ finalmente llega hasta el nodo destino, crea un paquete llamado *Route Reply* (RREP), que se comporta igual que el RREQ: es reenviado por la red hasta llegar al nodo origen.

De esta forma, al acabar el proceso, se habrán establecido dos rutas para llegar desde origen a destino, y de destino a origen.

El mantenimiento de las rutas también se tiene en cuenta en DYMO. Cuando un nodo recibe un paquete de datos que debe retransmitir hacia un destino al cual no tiene ruta conocida, notifica al origen mediante un paquete *Route Error* (RERR). Al recibir este RERR, el nodo origen sabrá que la ruta ya no es válida y la borrará.

4.2.3. DSR

DSR (*Dynamic Source Routing* [23]), es un protocolo para direcciones IPv4 que funciona totalmente bajo demanda, no usa en ningún momento el envío de paquetes de forma periódica, por lo que, este protocolo tiene una sobrecarga mínima.

Se basa en dos mecanismos que trabajan juntos, el de descubrimiento de rutas (*Route Discovery*) y el del mantenimiento de las rutas (*Route Maintenance).*

El primero de los mecanismos se encarga de descubrir la ruta del nodo origen al nodo destino solamente cuando se envíe un mensaje. Para ello envía en difusión un paquete *Route Request* y los nodos intermedios lo retransmiten de nuevo también en difusión, de esta forma habrá momentos en los que un nodo reciba más de una vez el mismo paquete, para que esto no sea un problema, estos mensajes llevan un número de secuencia que va aumentando en cada salto e indica cual es el que tiene la mejor ruta. Cuando el paquete llega a su destino, el receptor envía un *Route Reply* al origen por el mismo camino que el paquete ha llegado. Gracias a esto las rutas del emisor al receptor quedan descubiertas.

Cada nodo, además de guardar esta ruta, guarda en su caché otras rutas alternativas, para que en caso de pérdida de la ruta principal, se pueda invocar de nuevo al proceso de *Route Discovery* o poder utilizar una de las rutas alternativas. Estas rutas, también pueden servir para mejorar propiedades de la red, como el balanceo de carga o la robustez.

En el caso de que un nodo intermedio ya tenga la ruta en su caché al nodo destino, en vez de retransmitirla, envía un *Route Reply* al emisor con la información de la ruta.

El *Route Maintenance* se encarga de que mientras el nodo origen esté enviando algo al destino, si hay un cambio en la red o una pérdida de la ruta, el paquete sea capaz de tomar otro camino, pudiendo elegir entre activar el mecanismo de *Route Discovery* o usar una de las rutas almacenadas en caché.

Este protocolo, minimiza la sobrecarga, ya que no usa ningún tipo de mensajes periódicos, y es totalmente bajo demanda. Si es el caso de que ya se encuentran las rutas aprendidas por los nodos, la sobrecarga añadida es nula.

Si se tiene una topología de mucho movimiento de nodos donde se pierden rutas frecuentemente, la única sobrecarga que se crea, es solamente la que se produce al aprender de nuevo esas rutas perdidas, todos estos cambios no afectan a las que están actualmente en uso.

4.2.4. TORA

TORA (*Temporally-Ordered Routing Algorithm* [24]) es un protocolo especializado en grandes redes MANET, que a diferencia de la mayoría, no es un protocolo basado en vector distancia ni en estado del enlace, pertenece a un grupo llamado inversión del enlace (*link-reversal*). El protocolo crea diferentes rutas sin bucles para retransmitir el tráfico y puede funcionar como reactivo o proactivo.

TORA es un protocolo distribuido, los nodos guardan información solamente de sus vecinos. Cuando realiza un proceso reactivo, el protocolo va descubriendo las rutas siendo este proceso mucho más conveniente cuando hay grandes cambios de topología de la red, al mismo tiempo puede estar realizando el proceso proactivo, usando las tablas internas de cada nodo, este proceso puede ser más conveniente para rutas que son requeridas frecuentemente.

No usa rutas del camino más corto, algo inusual en la mayoría de los protocolos, lo que hace es fijar unas alturas a cada nodo, de forma que solamente se pueda ir de uno a otro si es de una altura mayor a una menor, y al ser rutas acíclicas se asegura que no pueda subir en ningún momento. No busca la ruta más óptima, si no que intenta minimizar la sobrecarga de la red.

Una de las principales ventajas de este protocolo, es el descubrimiento y la reparación automática de pérdidas de la ruta en la red, si una ruta desaparece, se hace un "*link-reversal*", que consiste en cambiar las alturas necesarias de los nodos para establecer un nuevo camino (siempre de alturas mayores a menores) hacia el destino.

4.3. Protocolos proactivos

4.3.1. DSDV

DSDV (*Destination-Sequenced Distance-Vector Routing* [21]) es un protocolo unicast proactivo adaptado del tradicional RIP (*Routing Information Protocol* [25]). Su principal objetivo es evitar los problemas de bucles en la actualización de las tablas de encaminamiento. Por lo cual añade un nuevo campo a las tablas RIP, el número de secuencia que permite distinguir entre una tabla antigua y una más reciente.

Como su nombre indica, DSDV implementa un algoritmo basado en vector de distancias. Eso significa que mantiene tablas con todos sus destinos accesibles junto con el siguiente salto, la métrica, y un número de secuencia de la entrada en la tabla generado por el nodo destino. Las tablas se mandan con mensajes de difusión de forma periódica o cuando ocurre un cambio significativo de la topología de red. Una ruta es considerada mejor que otra si tiene un número de secuencia mayor o, en caso de empate, si la distancia al destino es menor.

Cuando un nodo B detecta que la ruta hacia cierto destino D se ha perdido, inunda la red con una actualización de esa entrada en la que se incrementa el número de secuencia en uno y la distancia se marca como infinita.

Cuando un nodo A recibe este mensaje, incorpora a su tabla la actualización de la entrada hacia D a través de B siempre que no tuviera una entrada mejor para alcanzar D.

Para conseguir una cierta consistencia en las tablas de encaminamiento de cada nodo al cambiar la topología de la red, las actualizaciones deben ser frecuentes y suficientemente rápidas para que cada nodo pueda tener una visión realista de la red en un momento dado.

El problema fundamental de DSDV es la elevada sobrecarga de control que genera. Al no haber una especificación estándar, no hay productos comerciales basados en este protocolo.

Sin embargo, es la base sobre cual se han desarrollado otros protocolos como por ejemplo AODV.

4.3.2. OLSR

Las principales características de OLSR (*Optimized Link State Routing Protocol* [26]) son que se trata de un protocolo de encaminamiento proactivo con un mecanismo de inundación controlada.

Al ser un protocolo proactivo, se deben tener las rutas hacia todos los nodos constantemente actualizadas. Para descubrir los vecinos inmediatos, con los que hay conexión a nivel de enlace, se emiten periódicamente mensajes *HELLO*, que aparte de darse a conocer a los nodos con distancia de un salto, informan además de los vecinos inmediatos ya conocidos del nodo.

Una vez que se tiene conocimiento de los vecinos (y de los vecinos de éstos nodos, de distancia dos saltos), el nodo elige un conjunto de nodos para que actúen como sus *Multipoint Distribution Relay* (MPR). Estos nodos deben ser elegidos de forma que garanticen llegar a todos los nodos de dos saltos de distancia, de esta forma abrirán la ruta hacia cualquier nodo de la red.

Además, los MPR tienen la función de anunciar información sobre la topología de la red mediante inundación controlada, para que todos los participantes conozcan la ruta hacia el resto de la red. La inundación se realiza mediante mensajes *Topology Control* (TC), generados por nodos que han sido seleccionados como MPR, y que se reenvían de forma eficiente entre los MPR en lugar de hacerse mediante difusión masiva.

Para nuestro trabajo, se ha elegido este protocolo, se explicará en detalle más adelante.

4.3.3. TBRPF

TBRPF (*Topology Dissemination Based on Reverse-Path Forwarding* [27]) es un protocolo que provee un encaminamiento mediante el camino más corto posible, seleccionado mediante una modificación del algoritmo de Dijkstra.

Cada nodo tiene un árbol que almacena caminos a todos los nodos alcanzables basado en información parcial guardada en su tabla de rutas.

Para minimizar la sobrecarga, cada nodo solo transmite parte de su árbol a sus vecinos. Este protocolo combina una transmisión periódica con una transmisión diferencial (transmite solo si hay un cambio en la red)

El protocolo se diferencia en dos módulos independientes:

- ***Neighbor Discovery* (TND):** se basa en el intercambio de mensajes *HELLO* diferenciales, es decir, solamente contienen los cambios en el enlace, por lo tanto son más pequeños y se pueden enviar más frecuentemente y por lo tanto se detectarán los cambios en los vecinos mucho más rápidamente. Funciona a nivel de interfaces, es decir, cada interfaz enviará sus *HELLO* independientemente de las otras.

- **Modulo de encaminamiento:** cada nodo envía un subárbol del árbol de rutas mínimas que contiene, este subárbol se envía periódicamente cada un cierto tiempo y también se envía en actualizaciones diferenciales por cambios en la red, este método asegura la rápida actualización de la topología. Estos cambios del subárbol no son retransmitidos, si no que se espera a que llegue la siguiente transmisión periódica o diferencial para retransmitirlo.

4.2.4. FSR

FSR (*Fisheye State Routing Protocol* [28]) es un algoritmo que introduce el concepto de ámbito multi nivel (*multi-level scope*) para reducir la sobrecarga de la actualización de rutas en redes grandes.

Los nodos almacenan el estado del enlace para cada destino de la red y cada cierto tiempo los estados de los enlaces se actualizan por difusión a sus vecinos. La frecuencia de esta acción viene determinada por la distancia en saltos al destino, es decir, el ámbito relativo a cada uno.

Actualizaciones a nodos lejanos, que son propensos a ser menos usados, se actualizan menor número de veces que los cercanos, evitando una sobrecarga innecesaria.

En el viaje de un paquete, la ruta irá siendo más óptima según se acerque al destino.

4.4. Protocolos híbridos

4.4.1. ZRP

ZRP (*Zone Routing Protocol* [29]) es un protocolo de encaminamiento híbrido, ya que combina las mejores propiedades de los protocolos proactivos y reactivos.

ZRP se basa en separar la red en zonas. Se diferencian claramente una zona cercana o de vecindario, compuesta por los nodos que estén a un máximo de N saltos, y el resto de la red.

En ZRP se usan dos componentes para el encaminamiento. En la zona cercana se usa el componente *IntrA-zone Routing Protocol* (IARP) que actúa como un protocolo proactivo, manteniendo todas las rutas de los nodos que se encuentren a N saltos o menos, siendo N variable. El mecanismo usado para descubrir los nodos vecinos es el intercambio periódico de mensajes *HELLO*.

Para el encaminamiento global hacia los nodos fuera de la zona interior o cercana, ZRP cuenta con el componente *IntEr-zone Routing Protocol* (IERP), que se comporta como un protocolo reactivo.

Cuando se necesita la ruta hacia un nuevo nodo, usando el componente IERP se pide esta ruta con el mecanismo *Bordercast Resolution Protocol*

(BRP). Este mecanismo funciona mandando mensajes de petición de ruta a los nodos que pertenecen a la zona interior pero que además están en el borde, es decir, están al número máximo de saltos para ser considerados de la zona interior.

Si alguno de estos nodos del borde conoce la ruta, mandará un mensaje indicándosela al nodo que originó la petición. Si no es así, reenviarán la petición por toda la red hasta que llegue a un nodo que conozca una ruta hacia el destino. Entonces se enviará la respuesta de vuelta hasta el origen, guardando los nodos intermedios por los que pasa el mensaje para usarlos como ruta hacia el destino buscado.

Como se ha dicho antes, el radio (en número de saltos) de la zona interior es ajustable según las necesidades de la red. Como casos extremos tenemos que si este radio es pequeño, como mínimo uno, ZRP se comportará como un protocolo puramente reactivo. Si por el contrario el radio es infinito, el comportamiento será proactivo.

4.5. Comparación entre distintos protocolos de encaminamiento y elección de un protocolo adecuado para redes MANET.

Después de estudiar el comportamiento y los aspectos clave de los protocolos de encaminamiento más conocidos, se decidió usar OLSR como base para la sincronización del protocolo de autoconfiguración, por varios motivos.

El primero y más importante es que al ser un protocolo proactivo y con refresco de las rutas de forma periódica, nos da la seguridad de poder conocer la topología de la red en todo momento. Aunque algún paquete de actualización se pierda, las rutas se actualizarán igualmente con las siguientes transmisiones.

La otra gran ventaja que se notó es que OLSR aporta es la infraestructura creada para distribuir información a toda la red usando MPR, mecanismo que podría ser utilizado para hacer llegar nuevos mensajes de control a todos los nodos de la red.

Por último, el estar optimizado para funcionar en redes a gran escala también influyó en la decisión de utilizarlo, ya que se pretendía someter al protocolo de autoconfiguración resultante a pruebas exhaustivas en escenarios complejos.

Para terminar con el análisis de OLSR se intentó concretar qué información se podía deducir de este protocolo de encaminamiento, monitorizando su comportamiento o sus estructuras de datos; sin modificar ni su comportamiento ni sus paquetes. Se centró el interés en interpretar los datos que OLSR maneja, y los cambios que suceden con estos datos, como ideas de alto nivel.

Debido a que se tiene información actualizada sobre la topología de la red, podemos determinar la entrada o la salida de cualquier nodo de forma inmediata y sencilla. Además, se tendrá información acerca de esos nodos, como por ejemplo su IP.

Aunque parezca información escasa, más adelante veremos que será más que suficiente para desarrollar ideas más complejas.

También se vio que monitorizando las estructuras de datos que OLSR maneja se puede conocer el conjunto de nodos considerados vecinos, es decir, con los que se tiene un enlace directo. Dicho de otra forma, los nodos que se encuentran a un salto de distancia.

De nuevo, puede parecer información poco relevante, pero permitiría distinguir entre nodos vecinos o lejanos para controlar el comportamiento de mecanismos más complejos. Como ejemplo, esta información podría ser usada para decidir enviar ciertas peticiones sólo a los nodos vecinos, con el fin de tener respuestas inmediatas, o evitar inundar toda la red con mensajes de control.

El protocolo OLSR en detalle se explica en el siguiente capítulo.

4.6. Resumen

Un protocolo de encaminamiento de una red MANET tiene que satisfacer unos criterios que en otro tipo de redes no tienen lugar.

Existen diferentes tipos y diferentes clasificaciones, siendo la más extendida la que indica la forma de descubrir sus rutas.

Los resultados que nos ofrecen los protocolos reactivos y los proactivos son muy diferentes y dependiendo para qué caso convendrá elegir uno u otro.

Por ello es muy importante estudiarlos bien para saber qué protocolo se debe elegir para una red en concreto, observando la sobrecarga, latencia y demás factores que supondrán a la red.

5. Protocolo OLSR

5.1. Introducción

5.1.1. Qué tipo de protocolo es

El protocolo de encaminamiento OLSR [26] se deriva del *classical link state protocol* [30], pero adaptado para las redes MANET.

Se trata de un protocolo proactivo que, tal y como se ha explicado en secciones anteriores, significa que la información sobre las rutas hacia todos los nodos se mantiene siempre actualizada, para que esté disponible en caso de que sea necesaria.

Dentro de los protocolos proactivos, se pueden distinguir dos sub-tipos de protocolos según su comportamiento [31]: los conducidos por eventos (event driven), que envían paquetes con información sobre las rutas solo cuando éstas sufren algún cambio; y los que refrescan la información periódicamente (regular updated).

El protocolo OLSR entra dentro de la segunda categoría, ya que cada cierto tiempo paquetes con información sobre las rutas es retransmitida, aunque no se hayan detectado cambios.

La principal aportación de OLSR que lo diferencia de otros protocolos similares son las optimizaciones que se realizan para que la sobrecarga producida por las actualizaciones periódicas sea mínima. El modo en que se hace llegar la información sobre rutas a toda la red es mediante inundación controlada; designando a ciertos nodos encargados de enviarse entre ellos la información, haciéndola llegar al resto de la red, y comprobando que no se envían datos duplicados.

Debido a las optimizaciones que aplica, obtiene buenos resultados en redes grandes y densas. Sus optimizaciones se notan en el rendimiento cuanto más

grandes sean las redes, sobre todo si el tráfico es esporádico entre pares de nodos que varíen irregularmente, en lugar de comunicaciones regulares entre nodos concretos.

5.1.2. Breve descripción de su funcionamiento.

Esta sección trata del funcionamiento de OLSR, intentando resumir su *rfc*.

Lo primero que hace un nodo al iniciarse, es detectar con qué otros nodos tiene conexión a nivel de enlace. Para ello periódicamente se emiten mensajes *HELLO*. Estos mensajes no se retransmiten por los nodos que los reciben, ya que su finalidad es que los nodos se den a conocer a sus vecinos de un salto, es decir, nodos con los que existe conectividad a nivel de enlace. Otra funcionalidad de los mensajes *HELLO*, aparte de dar a conocer al propio nodo, es anunciar los vecinos ya conocidos del nodo emisor. De esta forma, un nodo que escucha estos mensajes no solo descubre a sus vecinos de un salto, sino que además adquiere conocimiento de sus vecinos de dos saltos de distancia.

La clave del protocolo está en los *Multipoint Relay*, cuyas siglas son MPR. Una vez que un nodo conoce el conjunto de sus vecinos de dos saltos, elige de entre sus vecinos de un salto un grupo de nodos MPR que retransmitan sus mensajes, de forma que le proporcionen acceso hacia todos los vecinos de dos saltos. Los nodos elegidos como MPR son notificados, ya que cada nodo mantiene información sobre quién le ha elegido como MPR en una estructura llamada *MPR selector set*.

Uno de los cometidos de los MPR es retransmitir los mensajes de difusión generados por alguno de los nodos en su *MPR selector set*. De este modo, los mensajes llegan a toda la red, pero intentando que la saturación sea mínima.

La otra tarea que debe realizar un nodo que ha sido seleccionado como MPR es generar y retransmitir mensajes *Topology Control* (TC), que dan a conocer al resto de la red los nodos de los que el emisor tiene constancia. Los mensajes TC se generan de forma periódica, y contienen una lista con las direcciones de los nodos del MPR selector set, nodos que han elegido al

emisor del mensaje como MPR, a diferencia de otros protocolos anteriores que anunciarían a cualquier nodo cercano. Con esto se consigue que la información sobre la topología generada sea mínima, y que su diseminación por la red se haga de forma controlada.

También es importante conocer las estructuras de datos internas que maneja OLSR. En particular, la más relevante es la tabla de rutas. La información que se tiene de cada nodo de la red en la tabla de rutas es una entrada con los siguientes campos: R_dest_addr R_next_addr R_dist R_iface_addr.

Esa entrada significa que el nodo identificado por la dirección R_dest_addr está a una distancia estimada de R_dist saltos, que el vecino con la dirección de interfaz R_next_addr es el siguiente salto en la ruta hacia R_dest_addr, y que este vecino es alcanzable a través de la interfaz local con la dirección R_iface_addr.

En OLSR se tiene en consideración la posibilidad de que un nodo tenga más de una interfaz de red participando al mismo momento en la red. Cada dirección de interfaz está asociada con una dirección principal, única para cada nodo. Esta dirección principal será la misma que la dirección de la interfaz en caso de que ese nodo tenga una única interfaz utilizando OLSR. Durante el resto de este capítulo no prestaremos mucha atención a los casos particulares a los que puede dar lugar este comportamiento y por lo general presumiremos que cada nodo tiene una única interfaz y dirección.

5.2. Formato del paquete OLSR

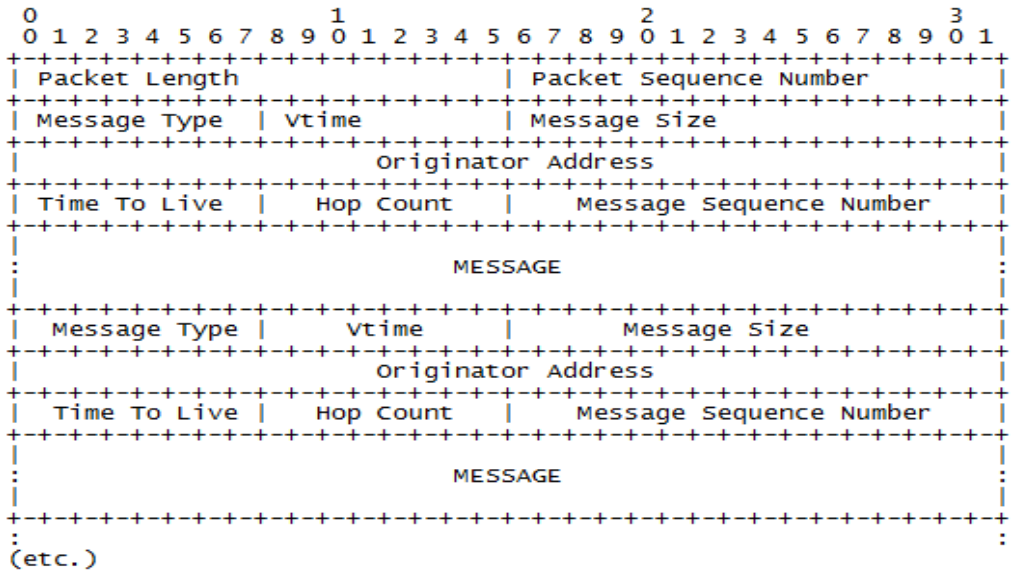
Las comunicaciones en OLSR se realizan usando un formato de paquete común para toda la información relacionada con el protocolo. De este modo, se facilitan futuras ampliaciones del protocolo sin romper la compatibilidad con versiones anteriores. Además, esto también facilita agrupar diferentes "tipos" de información dentro de una misma transmisión.

Los paquetes se encapsulan dentro de datagramas UDP para su transmisión por la red.

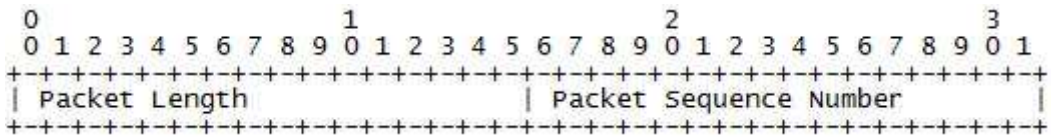
Cada paquete encapsula a su vez uno o varios mensajes. Los mensajes comparten un formato de cabecera común, lo que permite que un nodo sea capaz de aceptar y retransmitir (si procede) mensajes de tipo desconocido. Los mensajes pueden ser transmitidos por inundación a la red en su totalidad, o la transmisión puede ser limitada a nodos dentro de un determinado diámetro -refiriéndonos a número de saltos-, desde el emisor del mensaje. Por lo tanto, transmitir un mensaje al vecindario de un nodo es un caso especial de transmisión por inundamiento. Cuando se transmiten mensajes de control, las retransmisiones duplicadas son eliminadas de forma local, ya que cada nodo guarda información sobre los mensajes de control que ya ha transmitido anteriormente.

Los paquetes en OLSR usan el puerto UDP 698, asignado por el *Internet Assigned Numbers Authority* (IANA).

Los campos de cualquier paquete OLSR (omitiendo las cabeceras IP y UDP) son los siguientes:

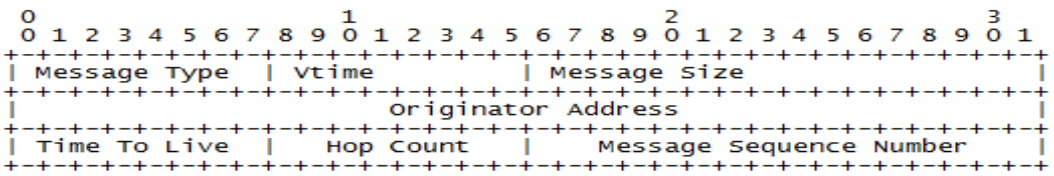


Cabecera del paquete:



- ***Paquet Length:*** Tamaño del paquete, en bytes.
- ***Packet Sequence Number***: Número de secuencia del paquete. Debe ser incrementado en uno cada vez que un nuevo paquete OLSR es transmitido.

Cabecera del mensaje:



- ***Message Type:*** Este campo indica qué tipo de mensaje se encuentra en el campo MESSAGE.
- ***Vtime:*** Campo que indica el tiempo de validez de la información contenida en el mensaje.
- ***Message Size:*** Tamaño del mensaje en bytes, incluyendo la cabecera y el campo MESSAGE.
- ***Originator Address:*** Dirección principal del nodo que originalmente generó el mensaje. No debe confundirse este campo con la dirección origen de la cabecera IP, que puede corresponder a la dirección del nodo que hace de intermediario.
- ***Time To Live:*** Contiene el número máximo de saltos que el mensaje será transmitido. Antes de retransmitir un mensaje, al valor de este campo se le debe restar uno.

- ***Hop Count:*** Número de saltos que el mensaje ha dado. Se inicializa a 0, y se incrementa en 1 en cada retransmisión.

- ***Message Sequence Number:*** Al generar un mensaje, el nodo que lo genera le asigna un número de secuencia único que identifica a cada mensaje en este campo.

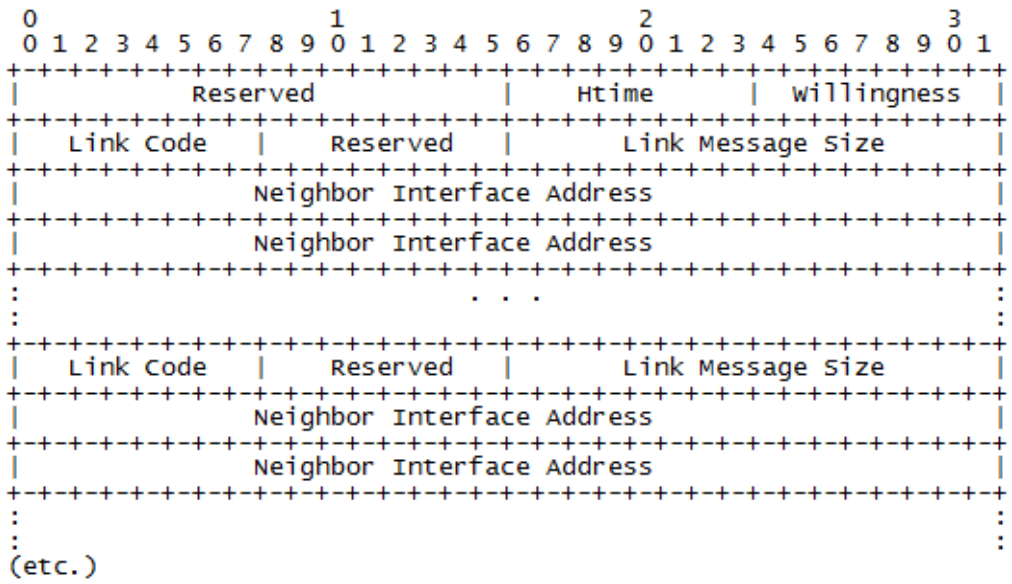
5.3. Mensaje HELLO

El protocolo OSLR utiliza el intercambio periódico de mensajes HELLO para descubrir a los nodos vecinos, y el estado de la red a nivel de enlace.

5.3.1. Formato del mensaje HELLO

Los mensajes *HELLO* siguen un formato similar al del paquete general, de modo que puedan incluir información para la detección del estado de enlaces de red, transmitir señales para la detección de nodos vecinos y selección de MPR, y tener en cuenta futuras ampliaciones.

Ese formato es el siguiente:



Se envía como datos dentro del paquete general OLSR descrito anteriormente, configurando el campo *Message Type* con el valor HELLO_MESSAGE, y el campo TTL con 1.

- ***Reserved***: Reservado, se establece con el valor "0000000000000".
- ***HTime:*** Este campo especifica el intervalo de emisión de mensajes *HELLO* usado por el nodo, esto es, el tiempo hasta el envío del próximo *HELLO*.
- ***Willingness***: Disposición o intención del nodo de retransmitir tráfico de otros nodos.
- ***Link Code:*** En este campo se coloca información sobre el estado del enlace entre el emisor y los nodos que se indican a continuación.
- ***Link Message Size:*** Tamaño del mensaje link, contado en bytes desde el principio del campo Link Code hasta el siguiente campo Link Code o, si no hay más mensajes link, el final de mensaje.
- ***Neighbor Interface Address:*** Dirección de un interfaz de un nodo vecino.

5.3.2. Procesamiento del mensaje HELLO

Los nodos procesan los mensajes *HELLO* recibidos para la detección de conexiones a nivel de enlace, detección de vecinos, y selección de MPR.

Todo esto se detalla a continuación.

5.4. Descubrimiento de vecinos

5.4.1. Detección de conexiones a nivel de enlace

Cada nodo guarda información sobre sus conexiones a nivel de enlace con otros nodos en una estructura llamada *Link Set*. Con estas conexiones nos referimos más concretamente a las interfaces de red utilizando OLSR y su capacidad de intercambiar paquetes OLSR.

El mecanismo usado para esta detección es el intercambio periódico de mensajes *HELLO*. Para considerar una conexión válida, se debe comprobar que existe comunicación (recepción de *HELLO*) en ambos sentidos.

Cada nodo vecino tiene asociado un estado en relación a la conexión: "simétrico" o "asimétrico". El primero indica que la comunicación en ambos sentidos ha sido confirmada; y el segundo se usa para indicar que se han recibido mensajes *HELLO* generados por el nodo vecino, pero no se ha confirmado aún que el nodo vecino es capaz de recibir los *HELLO* generados localmente.

La confirmación de que un nodo vecino es capaz de recibir los mensajes *HELLO* emitidos se tiene al encontrar la dirección propia en los *HELLO* del vecino.

5.4.2. Detección de vecinos

Cada nodo mantiene un conjunto de tuplas de vecinos (*neighbor tuples*) basadas en la información sobre las conexiones almacenadas en el *Link Set*.

Cada tupla de vecino consta de los datos (*N_neighbor_main_addr*, *N_status*, *N_willingness*), donde *N_neighbor_main_addr* es la dirección principal del nodo vecino, *N_status* se refiere al estado de la conexión (simétrica o asimétrica), y *N_willingness* es un entero entre 0 y 7 que representa la disposición o intención del vecino de retransmitir tráfico de otros nodos.

Aparte del conjunto de vecinos inmediatos o de un salto, con los que se tiene conexión a nivel de enlace, cada nodo guarda información sobre el conjunto de nodos de dos saltos de distancia, en la estructura *2-hop Neigbor Set*. Para ello, por cada nodo vecino de un salto, se guarda el conjunto de sus vecinos de un salto, ya que están anunciados en los mensajes *HELLO* que se reciben periódicamente.

5.4.3. Selección de MPR

Los MPR se usan para inundar mensajes de control de un nodo a toda la red minimizando las retransmisiones. Por lo tanto, el concepto de MPR se usa como una optimización del mecanismo habitual de inundación de mensajes en una red.

Cada nodo en la red elige, independientemente de los demás, su propio conjunto de MPR de entre sus vecinos de un salto con conexión simétrica. Cuando se ha elegido a un vecino como MPR, se anuncia en los mensajes *HELLO* colocando el valor *MPR_NEIGH* en lugar de *SYM_NEIGH* en el campo *Link Type* anterior a la dirección del vecino elegido como MPR.

El conjunto de MPR es calculado por cada nodo de forma que, a través de los vecinos escogidos, el nodo sea capaz de alcanzar a todos los vecinos de dos saltos. Más concretamente, a los vecinos de dos saltos exactos, por lo que no se está contando a los vecinos de un salto.

Aunque el conjunto de MPRs no debe ser mínimo para garantizar el correcto funcionamiento, cuanto más pequeño sea se consigue menor sobrecarga producida por los mensajes de control del protocolo OSLR.

Cada nodo guarda además en el *MPR selector set* el conjunto de nodos que le han elegido como MPR. Son detectados al procesar los mensajes *HELLO* recibidos.

5.5. Descubrimiento de la topología en OLSR

5.5.1. Funcionamiento

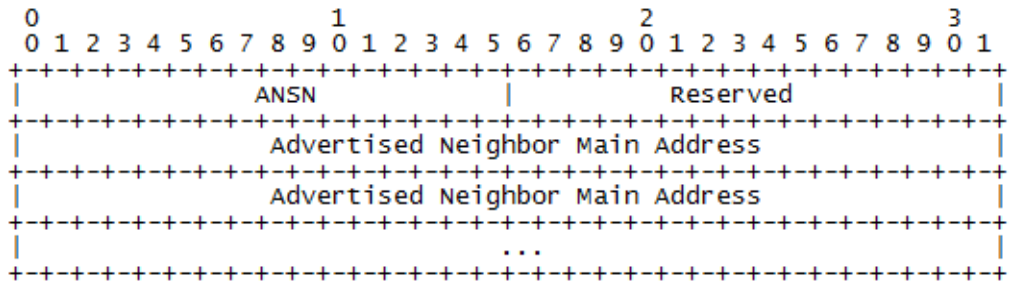
La detección de conexiones y vecinos proporciona a cada nodo una lista de nodos con los que comunicarse directamente y, haciendo uso de nodos MPR, un mecanismo para inundaciones optimizado. Basándose en esto, se genera información sobre la topología y se distribuye por la red.

Los mensajes *Topology Control* (TC) los generan los nodos que han sido elegidos como MPR por algún vecino suyo. Sirven para anunciar un conjunto de enlaces entre el emisor y otros nodos, que por lo general será su *MPR Selector Set*, es decir, los vecinos que han elegido al nodo emisor como MPR. Estos mensajes TC se emiten a toda la red por inundación.

La lista de direcciones anunciadas puede ser parcial en cada mensaje TC, por ejemplo, por limitaciones de tamaño de mensajes en la red; siempre que al unir varios todos los mensajes TC emitidos se encuentren todas las direcciones del *MPR Selector Set*.

5.5.2. Formato de los mensajes Topology Control

Los mensajes TC tienen el siguiente formato:



Esto se envía como datos dentro del paquete OLSR, configurando el campo *Message Type* con el valor *TC_MESSAGE*, y el campo TTL con el valor 255 (el máximo).

- ***Advertised Neighbor Sequence Number (ANSN):*** Un número de secuencia que se asocia con el conjunto de vecinos que se anuncia en este mensaje. Cada vez que un nodo detecta cambios en el conjunto de sus vecinos, se incrementa este número. Sirve para que los nodos que reciben este mensaje sepan si se trata de información más reciente que la que puedan tener actualmente.

- ***Advertised Neighbor Main Address:*** Campo que contiene la dirección principal de un nodo vecino.

- ***Reserved:*** Este campo está reservado. Se le da el valor "0000000000000000".

5.6. Cálculo de las tablas de rutas

Cada nodo mantiene actualizada una tabla con rutas hacia el resto de nodos de la red. Esta tabla se basa en la información obtenida sobre los nodos vecinos y los mensajes de control TC.

El formato de las entradas en esta tabla es el siguiente:

1. *R_dest_addr, R_next_addr, R_dist R_iface_addr*
2. *R_dest_addr, R_next_addr, R_dist R_iface_addr*
3. …

Cada entrada significa que el nodo con la dirección *R_dest_addr* se encuentra a una distancia estimada de *R_dist* saltos del nodo local, que el nodo vecino con la dirección de interfaz de red *R_next_addr* es el siguiente salto en la ruta hacia *R_dest_addr*, y que este nodo es alcanzable desde la interfaz local con la dirección *R_iface_addr*.

Se mantiene una entrada por cada nodo de la red para el que se tiene ruta conocida. Los nodos con una ruta desconocida no se incluyen. La actualización de la tabla se realiza en caso de aparición o desaparición de un nodo, ya sea vecino inmediato, vecino de dos saltos de distancia, o cualquier otro nodo conocido a través de los mensajes de control TC. También se actualiza la tabla cuando cambia información sobre las interfaces múltiples que puedan estar asociadas a los nodos.

Esta actualización de la tabla es un proceso interno, que no desencadena el envío de ningún mensaje.

6. Seguridad en redes MANET

Uno de los principales problemas de las redes MANET es su seguridad. La topología dinámica, el no tener ningún punto de infraestructura, nodos que entran y salen continuamente, otros que crean sus propias redes, constantes cambios en la distribución y en la naturaleza de los nodos, etc. hacen que estas redes sean vulnerables a distintos tipos de ataques.

Garantizar la seguridad en redes MANET es particularmente difícil, debido a la vulnerabilidad de los enlaces, la naturaleza esporádica de las conexiones, la ausencia de una autoridad certificadora (CA), el no existir un punto centralizado de monitorización, etc. Pero para que este tipo de redes puedan llegar a tener algún día éxito en el mundo comercial, es necesario tener algún sistema que nos ofrezca una red segura.

En este capítulo se hablará primero de las vulnerabilidades de las redes MANET, los tipos de ataques que estas redes pueden sufrir y los requisitos que una red segura debe cumplir. A continuación se explicará como funciona el sistema de administración de claves con ejemplos y distintos algoritmos que lo implementan. Por último se hablará de la gestión de confianza (monitorización) y de los encaminamientos seguros (integridad).

6.1 Vulnerabilidades

La naturaleza vulnerable de las redes MANET reside en el encaminamiento, en el uso de enlaces inalámbricos y en el mecanismo de autoconfiguración [32].

En el caso del encaminamiento, la vulnerabilidad depende de la confianza en que los nodos participantes entreguen información correcta de las rutas. Al tratarse de una comunicación multi-salto, un nodo intermedio malicioso puede enviar información errónea, modificar o eliminar el tráfico que pasa por él.

Incluso modificando parámetros de las rutas, podría llegar a controlar el tráfico de toda o una parte de la red.

El uso de enlaces inalámbricos hace que la red sea vulnerable a ataques pasivos, ya que un nodo malicioso con situarse dentro del rango de la señal de radio podrá escuchar e interceptar información privada.

Los mecanismos de autoconfiguración añaden aún más posibilidades de ataques, como la suplantación de identidad usando la misma dirección IP que otro nodo, el envío de mensajes de control falsos que gestionen maliciosamente la red, o la desestabilización de esta duplicando direcciones IP en uso.

Además, existen vulnerabilidades derivadas de las restricciones que tienen los dispositivos de este tipo de redes tales como la limitación de procesamiento, memoria o batería. Esto hace que un nodo malicioso pueda lanzar muy fácilmente un ataque de negación de servicio (*DoS*) a otro nodo ocupando toda su memoria o procesamiento, y por consiguiente forzar el agotamiento de su batería.

Ningún nodo o parte de la red está destinada a funciones específicas de seguridad (servicios administrativos, monitorización, autoridad de certificados), ya que no hay ninguna infraestructura. Pero incluso si se le asignara a algún nodo estas funciones, su disponibilidad no estaría garantizada debido a diferentes factores, como los cambios en la topología, particionado de la red o congestión en el área cercana de donde esté ese nodo con funciones.

La carencia de una infraestructura genera una ausencia de autoridad certificadora que establezca una línea de defensa entre nodos confiables y no confiables, que se distinguen por la posesión de un credencial. Para lograr esto en las redes MANET sería necesario que todos los nodos colaboraran, pero no se puede asumir que todos los nodos quieran colaborar o que lo hagan correctamente.

6.2 Ataques

Según se presenta en [33], se pueden distinguir dos tipos principales de ataques:

- **Ataques pasivos**: relacionados con el acceso a la información de un mensaje, como por ejemplo análisis del tráfico, escucha de datos o de claves, etc.

- **Ataques activos:** incluyen acciones como la modificación o borrado de información en los mensajes. Los ataques externos dirigidos a un objetivo en concreto forman parte de este grupo.

A continuación se listan los tipos de ataques en redes MANET:

- **Negación de servicio (DoS):** La forma tradicional de producir una negación de servicio en una red es inundar el recurso centralizado para que no pueda operar correctamente y así se desestabilice la red. Pero en las redes MANET, no hay ningún recurso centralizado. Una posible forma de conseguir una negación de servicio en un sistema distribuido es producir el agotamiento de las baterías o la saturación de la radio. Los nodos que reciben estos ataques tienen que ser capaces de reconfigurar su protocolo de encaminamiento.

- **Suplantación:** Si no existe un método de autenticación en la red, un nodo malicioso puede entrar en ella, enviar información de rutas y suplantar a algún nodo.

- **Revelación de la información:** La información transmitida en las comunicaciones puede ser escuchada y reutilizada posteriormente. Para evitar estos ataques esta información debe estar protegida con un algoritmo que la cifre.

- **Ataques de confianza:** Una jerarquía de confianza establece unos niveles de privilegio para cada nodo, de forma que un nodo no pueda suplantar a

otro de la red. Para conseguir esto es necesario un fuerte acceso de control que fuerce a cada nodo a respetar su nivel de privilegios en la jerarquía.

- **Ataques contra la seguridad de las rutas:** Se enmarcan dentro de los siguientes tipos:
  - **Ataques internos:** Consisten básicamente en enviar mediante difusión información errónea sobre las rutas.
  - **Ataques externos pasivos:** Son ataques que no modifican la información transmitida, solo la escuchan para obtener datos privados de la red como la posición, o las direcciones IP y MAC de los nodos.
  - **Ataques externos activos:** Tratan de modificar la información transmitida o congestionar la red mediante el envío de mensajes masivos o de suplantación. Su objetivo es modificar o incluso eliminar las rutas de los nodos.
- **Ataques militares:** Las redes usadas por militares necesitan la máxima seguridad. Los ataques que sufren se dividen en dos tipos:
  - **Ataques estratégicos:** ataques a la red enemiga con la intención de robar información para descubrir sus próximas acciones.
  - **Ataques tácticos:** Son más efectivos durante la batalla porque su objetivo es desestabilizar la red adversaria mediante, por ejemplo, la negación de servicio.

6.3. Requisitos de seguridad

Para que una red MANET sea segura necesita cumplir los siguientes requisitos [33]:

- **Disponibilidad**: Los servicios que tenga un nodo deben estar disponibles siempre, evitando en todo momento los posibles ataques.

- **Autenticidad:** Se debe asegurar que una comunicación se desarrolle entre el emisor y el receptor, evitando que nodos maliciosos suplanten a uno de ellos.

- **Confidencialidad:** Un nodo exterior a una conversación no puede tener acceso a la información transmitida.

- **Integridad:** Se ha de garantizar que un mensaje no sea modificado en algún paso intermedio de la comunicación.

- **Orden:** El envío de mensajes y las actualizaciones de los nodos deben realizarse en el orden correcto.

- ***Timeliness*:** El tiempo de actualización de rutas debe ser entregado dentro de un tiempo concreto.

- **Aislamiento:** La red debe ser capaz de aislar a nodos que se hayan detectado como maliciosos.

- **Autorización:** Cada nodo de la red debe tener una autorización generada por una autoridad certificadora.

- **Bajo nivel de computación:** Muchos dispositivos tienen limitaciones computacionales, de batería o de memoria, y esto debe tenerse en cuenta.

- **Localidad privada:** El protocolo de encaminamiento debe proteger la información sobre la localización de los nodos.

- **Auto estabilización:** La red debe ser capaz de recuperarse automáticamente de algún problema, no puede quedar deshabilitada.

- **Robustez bizantina:** Es una versión más estricta de la auto estabilización que exige también no cesar la funcionalidad de la red incluso cuando hay algún ataque en curso.

- **Anonimato:** Los nodos no deben mostrar ninguna de sus características (MAC, IP, etc.)

- **Gestión de claves:** El servicio de gestión de claves debe de crear un modelo de confianza proporcionando un criptosistema con creación, almacenamiento y distribución de claves.

- **Control de acceso:** Tiene que haber un control de acceso a la información.

- **Confianza:** Para que los usuarios quieran ser parte de la red, ésta debe mostrar confianza cumpliendo todos los requisitos de una red segura.

6.4. Mecanismos de seguridad

Los mecanismos de seguridad en una red MANET se basan en sistemas de claves, lo que permitirá un modelo de gestión de confianza en la red.

6.4.1. Administración de claves: PKI

El acrónimo PKI (*Public Key Infraestrucure*) hace referencia a un sistema complejo necesario para gestionar certificados y firmas digitales.

Permite a los usuarios autenticarse y poder usar claves públicas de los demás usuarios para cifrar o descifrar mensajes mediante algoritmos conocidos y debe cumplir las siguientes premisas:

- **Autenticación:** garantiza a un nodo que la identidad del nodo con el que se comunica es la correcta.

- **Confidencialidad:** la información no puede ser revelada a entidades que no estén autorizadas para ello.

- **Integridad:** la información no puede ser alterada sin autorización durante su transmisión.

- **No repudio:** un nodo no puede denegar acciones que realmente ha hecho. Por ejemplo, no puede negar el envío de un mensaje que si ha enviado.

- **Disponibilidad:** la red y sus servicios deben estar siempre disponibles aunque sufran ataques como la negación de servicio.

Un PKI se genera mediante un criptosistema basado en un sistema de claves y un algoritmo matemático con los que se permite cifrar un mensaje.

El cifrado consiste en aplicar dicho algoritmo a un mensaje para convertirlo en criptograma (texto ilegible) partiendo de la clave.

Podemos distinguir los siguientes tipos de cifrado:

- **Cifrado en flujo:** Usado en aplicaciones en tiempo real donde el cifrado se realiza continuamente bit a bit.

- **Cifrado por bloques:** Toma un bloque de un mensaje de un tamaño determinado y lo cifra obteniendo otro bloque del mismo tamaño.

Podemos distinguir dos tipos de criptosistemas dependiendo de las claves que se usen: simétricos y asimétricos.

I. Criptosistemas simétricos

Se trata de un sistema en el que ambos nodos participantes de una comunicación usan la misma clave, que es conocida por ambos de antemano. El mensaje cifrado viaja por la red, y un intruso será capaz de leerlo o modificarlo, pero no lo podrá entender. Si el atacante modifica el criptograma, las modificaciones no tendrán sentido, por lo que el receptor del mensaje lo detectará al descodificarlo.

El único requerimiento para este criptosistema es que los nodos conozcan la clave de antemano, y esto implica que la clave debe ser suministrada por un canal seguro, lo cual genera un nuevo problema: la distribución de claves de forma segura.

Es un tipo de sistema rápido con claves y criptogramas relativamente pequeños, pero no garantiza la seguridad por el problema de la distribución de la clave de una forma segura.

II. Criptosistemas asimétricos: Tipos y ejemplos

Suponen un gran avance en la criptografía ya que hemos visto que con los sistemas simétricos la seguridad se pierde si un atacante llega a conocer la clave.

Se basa en la existencia de un par de claves por cada nodo que dependen matemáticamente entre sí: una pública (conocida por todos los nodos) y otra privada (conocida solo por el propietario).

Cuando un nodo quiere enviar un mensaje lo cifrará con la clave pública del receptor, y éste lo descifrará con su clave privada al recibirlo.

En los sistemas simétricos se necesita un canal seguro para suministrar la clave. En los asimétricos esto no es necesario, pero si se necesita un canal de autenticación que asegure la identidad de la otra parte de la comunicación. Este sistema de autenticación se crea mediante certificados digitales. Se explicarán más adelante.

Un mensaje cifrado con una clave pública solamente puede descifrarse con su correspondiente privada. Estos pares de claves se obtienen mediante algoritmos basados en diferentes problemas matemáticos intratables que hacen que teniendo una clave pública sea imposible en un tiempo razonable obtener su privada. Por tanto si un atacante escucha los mensajes de la red, aunque pueda saber la clave pública no podrá obtener la clave privada, y por consiguiente no podrá descifrar los mensajes.

Existen diferentes tipos de criptosistemas asimétricos atendiendo al problema matemático intratable que los caracteriza:

- **Factorización:** Se basa en la dificultad de la factorización en primos de números enteros grandes como el algoritmo *RSA (Rivest Shamir Adleman)*.
- **Logaritmos discretos sobre grupos finitos:** Un logaritmo discreto es irresoluble en un tiempo razonable (*Diffie-Hellman, El Gamal*).

- **Logaritmos discretos sobre curvas elípticas:** En las matemáticas de las curvas elípticas, la resolución de un logaritmo discreto es aún más difícil, como *El Gamal Elíptico o KMOV (*Koyama Maurer Okamoto Vanstone*)* ya que trabajan sobre anillos, en lugar de grupos finitos.

Este sistema criptográfico, aunque es más seguro que el simétrico, presenta una serie de desventajas, como pueden ser el mayor tiempo de proceso y que las claves y el mensaje resultante cifrado son más grandes.

Además, el avance de la eficiencia de los nuevos algoritmos de factorización o de resolución de logaritmos discretos, está obligando a aumentar el tamaño de las claves.

Una solución a este problema son los criptosistemas basados en curvas elípticas, que nos proporcionan más velocidad y la misma seguridad con claves mucho más pequeñas y, por lo tanto, el cifrado resultante también es más pequeño.

Aun así existen sistemas más modernos creados en los últimos años, con los que podemos obtener mucha más velocidad de procesamiento que con las curvas elípticas, usando los criptosistemas basados en enrejado, como el criptosistema NTRU. Su velocidad radica en que solamente se basa en simples multiplicaciones polinomiales, y su seguridad está basada en la factorización de polinomios concretos en ciertos anillos.

RSA

RSA [34] es uno de los algoritmos más populares y extendidos. Fue inventado en 1977 por Ron Rivest, Adi Shamir y Leonard Adleman, de ahí su nombre.

Es un criptosistema de cifrado por bloques que basa su seguridad en dos problemas: el problema intratable de factorizar números grandes en primos y el problema RSA.

El algoritmo consiste en calcular el producto de dos números primos muy grandes (del orden de 10^{100}) elegidos aleatoriamente. Si se quisiera

comprometer la seguridad se tendría que llegar a estos dos primos. El método de obtenerlos es la factorización en números primos del resultado de la multiplicación mencionada, un problema irresoluble en un tiempo razonable.

Permite cifrar y descifrar mensajes y además puede generar claves de 1024-2048 bits, aunque actualmente se está proponiendo generar claves de 4096 bits por mayor seguridad.

En 1993, Meter Shor publicó su algoritmo para descifrar claves RSA demostrando que una computadora cuántica lo dejaría obsoleto. Por suerte, no se espera que las computadoras cuánticas acaben su desarrollo hasta dentro de muchos años.

Es un algoritmo generalmente lento en el que cobra gran importancia tener un buen generador de números primos aleatorios, porque si el atacante averigua qué primos se van a generar puede aplicar el algoritmo y obtener las claves públicas y privadas.

El Gamal

En 1985 se desarrolló El Gamal [35], un algoritmo basado en el problema intratable del logaritmo discreto. Parte de la idea del algoritmo Diffie-Hellman [36].

Puede ser utilizado tanto para cifrar o descifrar como para generar firmas digitales. Siendo la velocidad resultante para ello inferior a la obtenida con RSA pero con el inconveniente de que el resultado del cifrado es considerablemente largo.

Su encriptación es probabilística, esto quiere decir que un texto puede tener distintos cifrados.

A este tipo de algoritmos basados en el logaritmo discreto se les considera efectivos y seguros ya que no existen algoritmos suficientemente eficientes que en un tiempo razonable se puedan calcular esta operación.

El tiempo de cifrado y descifrado de El Gamal toma tiempos de $O(\log n)$.

Curvas elípticas

Los criptosistemas basados en curvas elípticas (CEE) aparecen como una alternativa a los criptosistemas antes mencionados, tanto por la disminución del tamaño de las claves que se requieren, manteniendo la misma seguridad computacional, como por el amplio abanico de grupos que ofrecen sobre el mismo cuerpo base. Su implantación en algunos sistemas de telecomunicaciones o tarjetas inteligentes es un hecho constatable, que parece aumentar día a día debido a las ventajas de estos criptosistemas [37].

Por ejemplo, una clave de 160 bits con CEE es tan segura como una de 1024 bits de RSA.

En los últimos años, este tipo de cripstosistemas han adquirido una gran importancia, llegando a formar parte de los estándares industriales.

Se han desarrollado variantes con curvas elípticas de los criptosistemas clásicos como RSA, pero su principal logro se ha conseguido con los criptosistemas basados en el logaritmo discreto como El Gamal, creando su nueva versión: El Gamal Elíptico.

Funciona bajo el problema intratable del logaritmo discreto, pero esta vez en vez de ser resuelto en grupos finitos, se resuelve en un dominio de puntos definido por un una curva elíptica más una operación aditiva formando un grupo abeliano.

Resolver un logaritmo discreto en estos puntos es mucho más difícil que en un grupo finito, por lo tanto se pueden usar claves mucho más pequeñas proporcionando la misma seguridad que con otros métodos.

NTRU

Propuesto a mediados de 1990, NRTU son una familia de criptosistemas para cifrar, descifrar, generar claves e incluso hacer firmas digitales. Fueron creados en 1996 por tres matemáticos de *Brown* (J. Hoffstein, J.Piper and J.H. Silverman)[38].

Son algoritmos basados en enrejados (*lattice-based*). Actualmente son mucho más rápidos que otros algoritmos como RSA o los basados en curvas elípticas, y además son más seguros.

Las operaciones de cifrado o descifrado se realizan en un tiempo $O(N^2)$ mientras que en otros algoritmos como RSA es de $O(N^3)$. Para la generación de claves en NTRU el tiempo necesario es de orden $O(N)$ siendo en otros de los algoritmos más rápidos de $O(N^2)$.

Sin embargo, una de las características que más llama la atención de este criptosistema es que, cualquier criptosistema mencionado antes, ya sea uno clásico o uno basado en CEE, se pueden resolver con una cantidad enorme de operaciones matemáticas que ordenador cuántico puede llevar a cabo, pero el criptosistema NTRU, se cree que es irrompible ante cualquier algoritmo, ni siquiera con un ordenador cuántico.

III. Certificado digital

Un certificado digital es un documento electrónico concedido por una autoridad de certificación que asocia una identidad y una clave pública, y dota al receptor de unas credenciales para operar en un determinado ámbito.

El certificado digital contiene una serie de campos como el identificador del receptor del certificado, su clave pública, tiempo de validez, etc. Además contiene una firma digital de la autoridad de certificación que permite comprobar la validez del certificado.

La utilidad de los certificados digitales se basa en la existencia de entidades confiables (CA) y en el conocimiento de su clave pública por parte de los clientes para poder verificar los cerificados de los demás.

El estándar más extendido es el X.509 [39], con los siguientes campos:

- Certificado
  - Versión
  - Número de serie
  - ID del algoritmo

  - Emisor
  - Validez
    - No antes de
    - No después de
  - Sujeto
  - Información de clave pública del sujeto
    - Algoritmo de clave pública
    - Clave pública del sujeto
  - Identificador único de emisor (opcional)
  - Identificador único de sujeto (opcional)
  - Extensiones (opcional)
    - ...
- Algoritmo usado para firmar el certificado
- Firma digital del certificado

IV. Firma digital

Una firma digital es un método criptográfico que asocia la identidad de una persona o equipo informático a un mensaje o documento electrónico.

Permite autenticar la identidad del emisor y verificar la integridad del mensaje o documento. Para proporcionar estos servicios se basa en la encriptación.

Para obtener la firma electrónica o *hash* de un documento (ver anexo), se deben seguir los siguientes pasos:

- Aplicar una función *hash* a los datos que queremos firmar.
- A la cadena binaria resultante se le aplica el algoritmo de cifrado. Para firmar se usará la clave secreta del firmante.
- Se añade la cadena resultante al documento.

Para verificar la firma, se siguen estos pasos:

- Se extrae la firma del documento y se obtiene la cadena *hash*, descifrando con la clave pública del emisor.
- Se aplica la función *hash* a los datos sin incluir la firma.

- Si ambas cadenas *hash* coinciden se acepta el documento.

A continuación se detalla un ejemplo de cómo se garantizan la autenticación y la integridad:

- **Autenticación:** Supongamos que A desea enviar un mensaje a C, suplantando a B. A desconoce la clave secreta de B, por lo tanto tendrá que firmar con otra clave, que se la podrá inventar o podría ser la suya propia. Cuando C aplique la clave pública de B para verificar la firma, la verificación fallará. Los datos que obtendrá no tendrán sentido o serán ilegibles.

- **Integridad:** Supongamos que A envía un mensaje a C a través de B. El nodo B modifica los datos enviados por A. Cuando C calcula la función *hash* de los datos de A, no coincidirá con la función *hash* descifrada de la firma. Por lo tanto descubrirá que los datos han sido modificados por el camino.

V. Autoridad de certificación

Una autoridad certificadora (CA) es una entidad de confianza que emite y revoca certificados digitales. Firma los certificados y confiando en esta entidad, se puede confiar en cualquier certificado generado por ella.

Además, puede tratar certificados que han pasado su periodo de validez, podrá renovarlos o revocarlos.

En la mayoría de las redes, el CA suele ser una entidad centralizada. En las redes MANET, al no tener esa entidad habrá que establecer una forma de crear CA a través de los nodos. Es decir, crear un modelo de gestión de confianza.

6.4.2. Gestión de confianza

Es necesario establecer un modelo de confianza para que la red sea segura. Un nuevo nodo deberá presentar sus credenciales para poder unirse y realizar la autenticación, mientras que en una comunicación interna un nodo puede verificar la identidad del otro.

La solución a esto puede ser un criptosistema [33], en el que los nodos manejan una clave pública y otra privada. Para desplegar un PKI es necesario una autoridad cerficadora (CA), es decir, una entidad responsable de crear certificados.

El uso de una CA centralizada en una red MANET puede ser un problema especialmente porque la entidad que la representa puede estar no disponible por una partición de la red, movimiento de nodos o un ataque de negación de servicio.

Otra forma de afrontar el problema es usar un grupo de K nodos que puedan certificar como un CA, pero de forma distribuída. El número de nodos elegidos será mayor que el necesario, por motivos de redundancia y seguridad ante posibles fallos. Para generar un certificado mediante difusión cada uno de estos K nodos crea una parte del certificado y, si se unen las suficientes partes, se podrá formar un certificado completo que se le entregará a un nuevo nodo.

Con esta posible solución, no se delega toda la responsabilidad en un único nodo, sino en varios. De esta forma, si alguno de estos nodos por algún motivo no está disponible, al existir más nodos de los que se requieren para poder formar un certificado, otro podrá generar su parte.

Además, la eficiencia de las operaciones de los CA puede ser mejorada si se combina con un encaminamiento seguro.

6.4.3. Encaminamiento seguro

Los protocolos de encaminamiento diseñados para redes con infraestructura y redes sin infraestructura tienen diferentes características.

Los de redes con infraestructura asumen la existencia de una entidad de confianza que participará asiduamente en el encaminamiento de los mensajes. En las redes MANET esto es muy distinto puesto que no existe esa entidad, sino que los mensajes tienen que pasar a través de los diferentes nodos de la red. Estos nodos no serán siempre los mismos ya que tienen movilidad, por lo

que deben poseer un certificado para asegurarnos de que no son maliciosos [32].

Por ello, los protocolos de encaminamiento tienen que ser específicamente diseñados para este tipo de redes.

Este campo ha experimentado una gran investigación en los últimos años. Entre otros, se ha propuesto una extensión del AODV para proteger los mensajes del protocolo de encaminamiento, llamado SAODV (*Secure AODV* [40]).

I. SAODV

En este protocolo se asume que todos los nodos conocen las claves públicas de los demás para que los nodos intermedios en una comunicación puedan ir validando el mensaje.

La idea se basa en que el nodo generador del mensaje añade a este una firma RSA y el último elemento de una cadena *hash*. Según el mensaje va atravesando la red, los nodos intermedios validan la firma RSA y la cadena *hash*, generan el elemento k-ésimo de la cadena *hash*, siendo k el número de saltos, y lo añaden al mensaje.

6.5. Resumen

Las redes MANET sufren una serie de vulnerabilidades que otras redes no tienen, por lo que pueden recibir ataques de diversos tipos. Para poder crear una red segura, es necesario adoptar nuevas ideas y protocolos ya que los de las redes cableadas o inalámbricas con infraestructura son insuficientes para cumplir los requisitos que una red MANET tiene.

Una posible solución es el uso de un sistema de claves, que nos permitirá establecer una gestión de confianza entre los nodos de la red. Además, será necesario que el protocolo de encaminamiento sea seguro y se adapte a las características que una red MANET con seguridad supone.

Un sistema de claves, necesitará la existencia de certificados digitales con una firma digital y un criptosistema que podrá ser simétrico o asimétrico.

Los simétricos necesitan de un clave conocida previamente que tiene que suministrarse por un canal seguro, por lo que estos sistemas no son muy fiables. Sus ventajas son que se trata de sistemas rápidos y usan claves y criptogramas cortos.

Los asimétricos nos garantizan una seguridad mayor al usar además claves privadas, pero son más lentos y generan claves y criptogramas más largos. Las claves pública y privada de un nodo están relacionadas matemáticamente mediante algoritmos basados en algún problema intratable, de forma que sea imposible en un tiempo razonable obtener la clave privada partiendo de la pública.

BLOQUE II

Tras el estudio llevado a cabo para conocer el estado actual del área de investigación relativo a los protocolos de autoconfiguración para redes MANET, una de las propuestas existentes se consideró especialmente interesante. Esta propuesta de protocolo de autoconfiguración para redes MANET se encuentra en el artículo titulado "*IP address assignment in a Mobile ad hoc network*", publicado por Mansoor Mohsin y Ravi Prakash en 2002 [9]. Esta línea de investigación ha sido continuada por otros investigadores que han propuesto mejoras al protocolo original. Una de estas investigaciones se llevó a cabo en 2006 por los alumnos Miguel Ángel Tolosa Diosdado y Adam Ameziane, en el trabajo de sistemas informáticos con el título "Sistema de autoconfiguración para redes Ad Hoc"[38].

El protocolo descrito en el trabajo de estos alumnos define una serie de conceptos interesantes. Pero estudiándolo en profundidad, puede verse que ciertos aspectos son mejorables, y otros no están todo lo bien definidos que podrían. Por ejemplo, como mecanismo de sincronización se propone el envío de numerosos mensajes entre todos los nodos de la red.

En el planteamiento inicial, se fijó como meta para este proyecto estudiar el protocolo mencionado y proporcionarle un mecanismo de sincronización más desarrollado, robusto y completo; y estudiar la forma de proporcionar la base para la creación de un sistema de seguridad que respondiese a las necesidades de las redes MANET, basando su funcionamiento en la autoconfiguración.

El trabajo realizado se puede diferenciar por lo tanto en dos contribuciones principales: la creación de un protocolo de autoconfiguración, SARA, y el desarrollo de un módulo de seguridad para éste.

Con respecto al protocolo de autoconfiguración anteriormente citado, tras un estudio preliminar del funcionamiento se vio que aparte de reemplazar el mecanismo de sincronización el protocolo podía mejorarse en otros aspectos

más fundamentales. Más adelante se verá cómo se puede simplificar el mecanismo que determina cómo se recuperan direcciones que quedan disponibles al salir un nodo de la red, haciéndolo más eficiente y facilitando usar una sincronización menos compleja.

Por ello, lo que se propone en esta memoria no es un simple añadido al protocolo de partida, sino una nueva versión mejorada. Aunque claramente conserva las ideas de fondo, se ha replanteado completamente la forma de llevarlas a la práctica.

También cabe destacar que desde un principio se tuvo presente que el protocolo no sólo debía parecer factible, sino que debía demostrar su correcto funcionamiento una vez implementado en el entorno de simulación NS-3.

El hecho de que se fuese a implementar hizo que se prestase especial atención a cualquier pequeño detalle que pudiese hacer que el protocolo fallase, por lo que se han tenido en cuenta multitud de casos extremos o poco frecuentes durante su planteamiento.

7. Protocolo original de autoconfiguración

7.1. Introducción

El trabajo de partida se trata de un protocolo de autoconfiguración para redes MANET que garantiza la unicidad de las direcciones IP usadas en la red usando como base la idea de la división binaria de direcciones.

En este protocolo, todos los nodos de la red son los encargados de asignar direcciones IP a los nuevos nodos que entran. Cada uno tiene asignadas una serie de direcciones IP que puede asignar a los nuevos nodos. También es responsabilidad de todos los nodos el mantener actualizada la información sobre las direcciones IP libres disponibles propias y del resto de nodos de la red. Mediante un sistema de sincronización simple, los nodos son capaces de mantener las direcciones IP asignadas únicas, garantizando la no duplicidad.

El protocolo especifica cómo actuar en los sucesos de entrada o salida de nodos de la red, así como la partición y la unión de redes.

7.2. Estructuras de datos

Cada nodo debe mantener dos tablas que se irán actualizando constantemente mediante un mecanismo de sincronización para poder reflejar el estado de la red actual. Recordemos que la topología dinámica de este tipo de redes conlleva sucesos como la entrada y salida de nodos, o la partición y unión de redes.

- ***Free_IP_blocks*:** tabla que contiene los bloques de direcciones IP libres para atender a los nuevos nodos. Cada bloque almacenado debe cumplir la condición de ser disjunto del resto de bloques de la red. En esta tabla también se guardará la dirección IP del propio nodo**.**

- ***Allocated_IP_blocks****:* tabla que almacena la topología de la red. Cada entrada de la tabla se refiere a un nodo, guardando los siguientes datos*:*
    - ***Dirección IP:*** asignada a la interfaz.
    - ***Free_IP_blocks***: bloques de direcciones IP libres indicando en qué bloque se encuentra la dirección IP de la interfaz.
    - ***TS (TimeStamp****):* indica la antigüedad del nodo, necesaria a la hora de recuperar direcciones que algún nodo deja disponibles al abandonar la red.
    - ***Father:*** Dirección IP del nodo que hizo de servidor cuando entró a la red.

Además cada nodo debe conocer el identificador de la red PID (*Partition ID*), generado por el nodo que creó la red. Este PID es necesario en la partición y unión de redes.

7.3. Formato de los mensajes

En la siguiente tabla se listan los mensajes de este protocolo y su funcionalidad:

MENSAJE	DESCRIPCIÓN
ADDR_REQ	Mensaje que el nodo cliente envía al servidor para solicitar una dirección IP.
ADDR_REP	Respuesta del servidor al nodo cliente cuando recibe el mensaje de solicitud de dirección IP.
SERVER_POLL	El nodo cliente envía ese mensaje a un nodo servidor para indicarle que lo escogió como servidor para la configuración de su dirección IP.

IP_ASSIGNED	Mensaje enviado por el nodo servidor al nodo cliente asociando la dirección IP.
IP_ASSIGNMENT_OK	Mensaje enviado por el nodo cliente al nodo servidor confirmando que el proceso de autoconfiguración fue realizado con éxito.
NODE_UP	Mensaje enviado para notificar la entrada de un nuevo nodo, de forma que el resto de ellos puedan actualizar las tablas.
NODE_UP_REPLY	Mensaje de confirmación de actualización de tablas tras la entrada de un nodo.
GRACEFULL_DEPARTURE	Mensaje enviado para entregar las direcciones libres que deja un nodo al salir avisando de la red.
NODE_DOWN	Mensaje que se envía a todos los miembros de la red para notificarles la desaparición de un nodo.
NODE_DOWN_REPLY	Mensaje de respuesta al Node_down para indicar que los cambios en las tablas han sido realizados.

7.4. Funcionamiento general

Cada nodo guarda en sus estructuras de datos, entre otros, una serie de bloques de direcciones IP libres. Estos bloques son la base fundamental en la que se sostiene el mecanismo de asignación de direcciones por división binaria.

La figura 7.1 muestra cómo se realiza la división binaria de esos bloques de direcciones libres, partiendo de un nodo que creó la red con 256 direcciones IP.

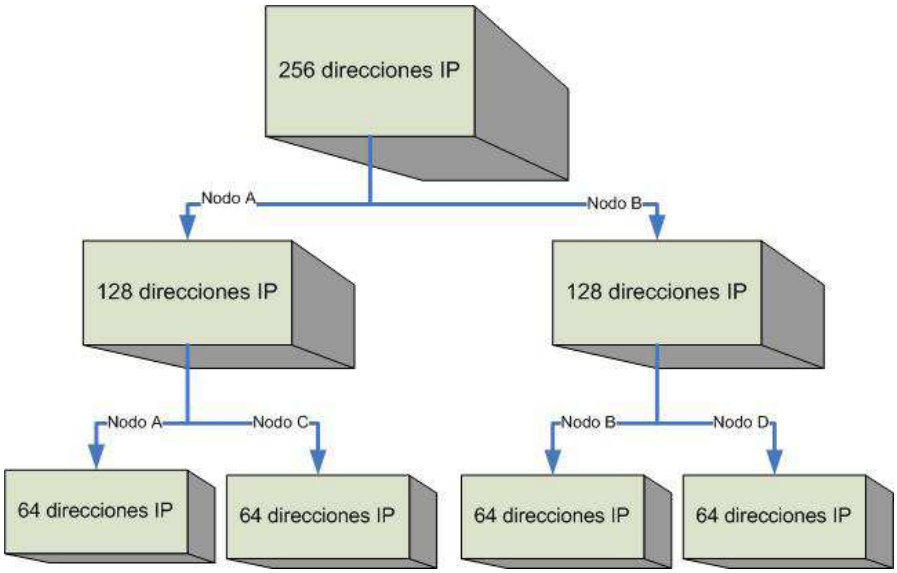


Figura 7.1 – Funcionamiento del modelo de división binaria.

Inicialmente, el nodo A posee todo el rango de direcciones IP, menos una que está asociada a su interfaz de red, por lo que tiene un bloque de 255 direcciones IP libres para entregar a otros nodos.

Cuando el nodo A recibe una solicitud de entrada a la red del nodo B, divide su conjunto de direcciones IP libres en dos mitades, suministrando la mitad para el nodo B y quedándose con la otra mitad. Ahora, el nodo A y el nodo B tienen cada uno 128 direcciones, siendo una dirección IP para su interfaz de red y 127 para servir a otras peticiones.

Usando el mismo mecanismo, el nodo A divide su conjunto de direcciones IP libres en dos mitades nuevamente, suministrando la mitad para el nodo C y quedándose con la otra mitad. El mismo procedimiento ocurre con el nodo B que atiende una petición del nodo D.

El mecanismo de división binaria asegura que todos los nodos de la red posean conjuntos disjuntos de direcciones IP, evitando que una misma dirección IP pueda ser usada por dos o más nodos aún cuando se produzca la unión de dos redes ad hoc. Es decir, asegura completamente la unicidad de las direcciones IP.

Tomando como base este modelo de distribución de direcciones, se muestran a continuación todos los escenarios que pueden tener lugar teniendo en cuenta la topología dinámica de las redes MANET.

7.4.1. Inicialización de la red MANET

La inicialización de un nodo consiste en el intento de conexión a una red y, si esta no existe, en la creación de la suya propia.

Para asociarse a una red un nodo envía mensajes ADDR_REQ (*address_request*) solicitando una dirección IP y un bloque de direcciones libres. Si en un número definido de intentos no recibe respuesta alguna, creará una red propia y generará un PID que quedará asociado a la nueva red. De este modo se convertirá en el nodo líder que poseerá todas las direcciones libres de la red. Para su interfaz reservará la primera dirección.

7.4.2. Entrada de nodos

Cuando un nodo desea unirse a la red, con el objetivo de obtener una dirección IP envía un mensaje ADDR_REQ por difusión. Estos primeros mensajes se envían usando la capa MAC, ya que no se dispone aún de una dirección IP para el nuevo nodo. Cualquier nodo perteneciente a la red responde a la petición del nodo cliente enviando un mensaje ADDR_REP (Address Response). Este mensaje contiene el mayor bloque free_ip_block de entre los que dispone, ya que el nodo puede poseer más de uno con diferente número de direcciones IP.

El nodo cliente puede recibir más de un mensaje proveniente de diferentes nodos, y elegirá como nodo servidor al que le haya enviado el mayor *free_ip_block* enviándole un mensaje SERVER_POLL.

Una vez se haya recibido el mensaje SERVER_POLL del nodo cliente confirmando la intención de obtener una dirección IP, el nodo servidor divide su *free_ip_block* por la mitad, suministrando una mitad al nodo cliente y quedándose la otra mitad para atender futuras peticiones.

Esta mitad del bloque es enviada a través del mensaje IP_ASSIGNED. El nodo cliente asocia la primera dirección IP de este bloque recibido como su dirección de red, y marca al bloque como el *free_ip_block* que almacena su propia dirección IP.

Para confirmar que la configuración fue realizada con éxito, el nodo cliente envía un mensaje IP_ASSIGNMENT_OK al nodo servidor.

A continuación informa a cada uno de los nodos de la red su presencia en la misma enviando un mensaje *NODE_UP* a cada uno las veces que sean necesarias, hasta recibir las correspondientes respuestas de confirmación *NODE_UP_REPLY* de cada nodo. Estos mensajes ya se envían mediante datagramas IP.

El nodo cliente establecerá en sus estructuras de datos internas como *father* al nodo que le entregó el *free_ip_block*.

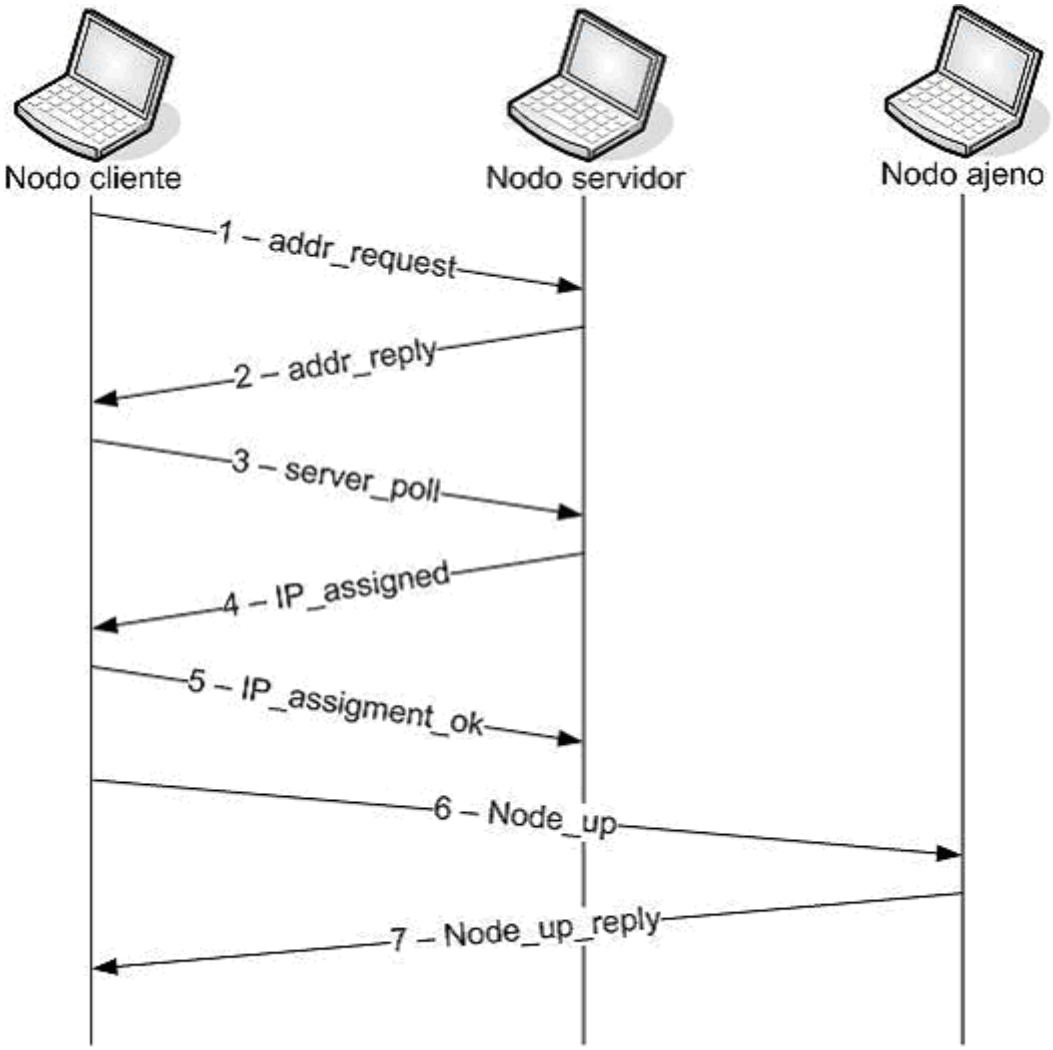


Figura 7.2 – Entrada de un nodo en la red

La figura 7.2 muestra el paso de mensajes durante el proceso de la entrada de un nodo en la red.

Puede darse el caso de pérdidas de mensajes en el proceso de entrada de los nodos. Si el nodo servidor no recibe el *IP_ASSIGNMENT_OK*, manda un mensaje *ping* al nodo cliente para verificar que fue definitivamente configurado. Si recibe respuesta, eso implica que el proceso de autoconfiguración fue realizado con éxito Por lo tanto, el mensaje de confirmación se perdió.

Por otro lado, si el nodo cliente no recibe algún mensaje *NODE_UP_REPLY*, comprobará que los nodos siguen aún en la red mediante el envío de mensajes *ping*, y si es así reenviará el mensaje *NODE_UP*, hasta recibir respuesta.

Es posible que un nodo servidor no tenga ninguna dirección IP disponible en su *free_ip_block* al recibir un mensaje de petición de un nodo entrante. La solución es encaminar la petición hacia los nodos vecinos en la red. Para mantener una distribución uniforme de las direcciones IP, el nodo servidor mira en su tabla *allocated_ip_blocks* para buscar cuál es el nodo que posee mayor número de direcciones libres.

Si ninguno de los nodos posee una dirección IP libre, el nodo servidor responde al cliente enviando un mensaje *DENY* informando que no hay direcciones IP disponibles en ese momento.

7.4.3. Salida de nodos

La salida de nodos en este protocolo se trata de dos maneras diferentes, diferenciando entre salida fácil y brusca. Una salida brusca está causada por ejemplo por la desconexión de un nodo, el agotamiento de la batería o simplemente el desplazamiento hacia una zona fuera del rango de cobertura de la red.

Salida fácil:

Cuando un nodo desea abandonar la red lo notifica facilitando la tarea de recuperar las direcciones que dejará libres.

Para realizar la notificación comprueba si su padre se encuentra en la red. Si su padre responde a mensajes *ping,* entonces le envía un mensaje *GRACEFUL_DEP* y abandona la red.

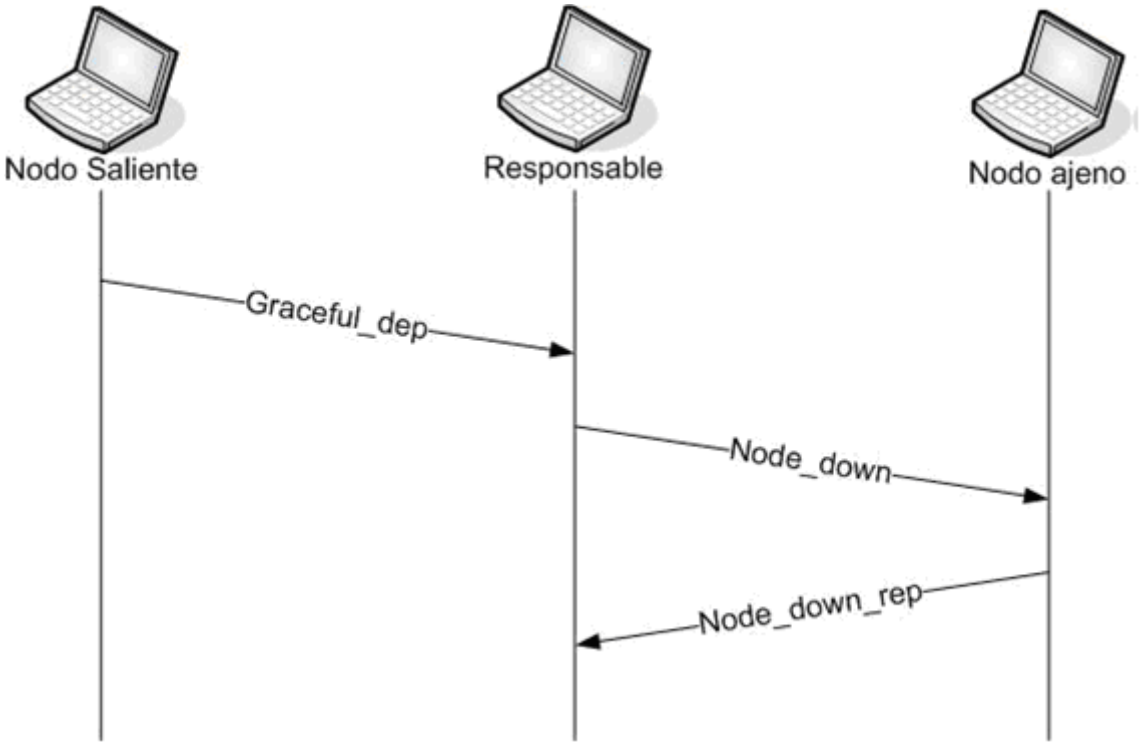


Figura 7.3 - Proceso de salida fácil de un nodo.

Si su padre no se encuentra en la red, el nodo responsable de recoger sus direcciones es el nodo con mayor antigüedad. Por lo tanto, es a él a quien envía el mensaje *GRACEFUL_DEP*.

Tanto si es el padre como si es el nodo de mayor antigüedad, el nodo responsable recoge las direcciones IP y actualiza su tabla con los nuevos cambios. Después manda un mensaje *NODE_DOWN* a todos los nodos de la red avisando que un nodo ha abandonado la red. De este modo todos actualizan las tablas, y para confirmarlo responden con el mensaje *NODE_DOWN_REP* Si algún nodo no responde con este mensaje, se reenvía el mensaje *NODE_DOWN* hasta que todos los nodos tengan constancia del cambio.

Si se pierde algún mensaje de salida, se considerará salida brusca.

Salida brusca:

Si un nodo sale de una forma brusca no tiene ocasión de avisar a ningún otro de su salida. Por lo tanto sus free_ip_blocks se perderían si no se tuviese un mecanismo diseñado para estas situaciones.

Los nodos comprueban periódicamente que su nodo padre y sus nodos hijos siguen en la red mediante mensajes *ping*.

Si los nodos hijos han abandonado la red, el nodo recupera las direcciones que le asignó.

Por otra parte, si es su nodo padre el que ha abandonado la red, el nodo se encargará de sus direcciones solo si es el nodo más antigüo de la red. Notificará mediante mensajes *NODE_DOWN* la caída del nodo, para que el resto de integrantes de la red mantengan sus tablas actualizadas.

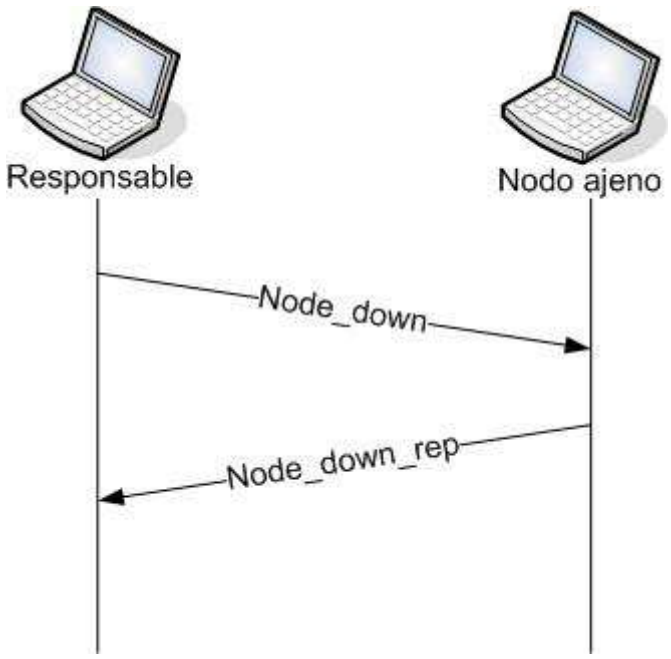


Figura 7.4 – Intercambio de mensajes en la salida brusca de un nodo.

7.4.4. Sincronización

Para que todos los nodos tengan actualizadas las tablas que representan el estado de la red, es necesario un proceso de sincronización que permita detectar los cambios producidos en esta.

Cada nodo dispone de una tabla *Allocated_ip_block*, que contiene información de todos los nodos de la red: los padres de cada uno, sus direcciones libres, y su TS. Tiene que ser igual en todos los nodos, y ante posibles cambios en la red, debe ser actualizada en el menor tiempo posible.

En el caso de entrada de nodos en la red la sincronización se realiza enviando los mensajes *NODE_UP* y *NODE_UP_REPLY*.

Cuando se produce una salida fácil se informa de la salida mediante el mensaje *GRACEFUL_DEP,* y el nodo que recogerá sus direcciones IP avisará al resto de nodos mediante el mensaje *NODE_DOWN*.

Si se da el caso de que la salida es brusca, también puede detectarse ya que se dispone de un temporizador de sincronización para que cada nodo realice una comprobación de los nodos de los que es responsable cada cierto tiempo. Cada nodo se responsabiliza de los nodos a los que ha entregado direcciones libres, y de su nodo *father*. En caso de que un nodo no tenga a otro responsable de él, lo será el nodo más antiguo de la red. Si en una de estas comprobaciones mediante mensajes *ping* un nodo comprueba la desaparición de otro, realizará la sincronización notificando al resto de la caída del nodo.

7.4.5. Partición y fusión de redes

El protocolo soporta la partición y fusión de redes, pero es un tema en el que no se va a entrar en detalle debido a que el protocolo SARA que se realizó en este proyecto no contempla esta funcionalidad como una de sus metas.

El principio que rige este funcionamiento es el que cada red tiene un PID. Cuando un subgrupo de la red la abandona, esto es, se produce una partición, el mecanismo de actualización detectará el cambio como si se tratase de varias salidas bruscas al mismo tiempo. Por lo tanto, la sincronización está resuelta en el caso de división de redes.

Al fusionarse dos redes, se dispone de un mecanismo que intenta resolver los conflictos que se puedan dar (direcciones duplicadas). Se basa en el PID y el campo TS de cada nodo para detectar cuando dos nodos diferentes comparten dirección IP.

8. Protocolo SARA

8.1. Ideas aportadas en el nuevo protocolo

8.1.1. Introducción

En esta sección expondremos cómo se ha mejorado el protocolo anterior optimizando sus estructuras de datos y la forma en la que se recuperan bloques de direcciones IP libres.

También se hablará del nuevo proceso de sincronización, usando el protocolo OLSR y de la inicialización de las estructuras de datos a la hora de entrar un nodo en la red.

Por último, se razonarán las dificultades a los que los nuevos mecanismos propuestos deben hacer frente.

El objetivo de esta sección es desarrollar las ideas sin entrar en demasiados detalles técnicos que obstaculicen su explicación. De modo complementario, la sección 8.2 está dedicada a la especificación formal del protocolo SARA.

8.1.2. Mejoras sobre el protocolo original

En el protocolo de partida se tiene que por cada nodo de la red hay que conocer la siguiente información: su dirección IP, los bloques de direcciones libres *Free_IP_blocks*, el nodo *father* que le suministró la dirección IP, y su marca de tiempo TS.

Recordemos que *Free_IP_blocks* se refiere a varios bloques disjuntos de direcciones IP, del tipo [.5 - .10], [.15 - .20], [.48 - .120]

Los campos *father* y TS se usan para determinar quién es el responsable de recuperar las direcciones libres que un nodo deja disponibles al abandonar la

red. Este responsable es el nodo *father*, o en caso de no estar disponible, el nodo de la red con menor TS.

El hecho de que cada nodo pueda tener muchos bloques diferentes de direcciones libres hace que no se pueda determinar cuánto puede llegar a ocupar el campo *Free_IP_blocks*.

Se vio que si se cambiase el modo en el que se recuperan las direcciones de un nodo que sale de la red, los campos *father* y TS podrían omitirse.

Las diferencias de este nuevo mecanismo con respecto al original se resumen en los siguientes puntos:

- Cada nodo tiene como información asociada para la autoconfiguración un único bloque de direcciones IP libres contiguas, que incluyen a la propia dirección IP del nodo.

- El responsable de recuperar las direcciones IP que un nodo que abandona la red deja disponibles es aquel que puede unir por la derecha ese bloque libre con el suyo.

  Esto no será posible cuando el bloque que se debe recoger contiene la dirección más baja de la red. En ese caso, el nodo que recoge el bloque es aquel que puede añadirlo al suyo por la izquierda.

- Al dividir las direcciones libres en dos bloques para entregar uno de ellos a un nuevo nodo que entra a la red, el nodo que hace de servidor entrega al cliente el sub-bloque que no contiene su propia dirección IP.

Con este nuevo comportamiento conseguimos varias mejoras.

Por una parte ya no son necesarios los campos *father* y TS, ya que el propio bloque de direcciones libres de cada nodo determina quién es el responsable de recogerlo en caso de que un nodo abandone la red.

Se garantiza además que cada nodo tiene un solo bloque de direcciones IP libres en todo momento, haciéndolo más manejable. Se conoce cuánto ocupará esta información, y se garantiza que este tamaño será constante. Esto es

importante para poder estudiar los requisitos de memoria de los nodos, o la sobrecarga que supondría su transmisión por la red.

En el protocolo original, cada nodo debe comprobar periódicamente mediante mensajes *ping* si siguen activos tanto el nodo "padre" como sus "hijos". El nodo padre es el que le suministró su dirección IP, y los nodos hijos son los nodos a los que se les ha facilitado una dirección IP y un bloque de direcciones libres.

Con las mejoras propuestas, se consigue que los nodos no tengan que comprobar activamente si otros nodos siguen participando en la red. Será en el momento de detectar que otro nodo ha abandonado la red, cuando cada nodo comprobará si el bloque de direcciones que queda disponible debe ser recogido por él mismo, o por otro de los nodos de la red.

En el siguiente apartado veremos un ejemplo que muestra este comportamiento de forma más clara.

8.1.3. Sincronización usando OLSR

La solución que proponemos a la necesidad de un mecanismo de sincronización para el protocolo SARA es integrar en parte el protocolo de encaminamiento OLSR.

Al integrar la autoconfiguración con OLSR, conseguimos aprovechar el mecanismo provisto por este último para detectar los cambios en la red. Recordemos que OLSR se trata de un protocolo de encaminamiento proactivo, y que mantiene información actualizada sobre las rutas hacia todos los nodos de la red. Por ello, es capaz de detectar la entrada de un nuevo nodo a la red (descubrimiento de una ruta nueva), y la salida de nodos (eliminación de una de las rutas conocidas).

Por cada nodo de la red, OLSR mantiene una entrada en su tabla de rutas. Esta tabla consta de los campos *R_dest_addr R_next_addr R_dist R_iface_addr*. Podemos ver un ejemplo de esta tabla en la siguiente figura, en la que omitimos los campos referentes a las rutas por simplicidad:

R_dest_addr	R_next_addr	R_dist	R_iface_addr
.1	…	…	…
.128	…	…	…
.65	…	…	…

En el protocolo SARA necesitamos conocer el bloque de direcciones IP que tiene asignado cada nodo de la red, de modo que cada nodo tendrá almacenada una tabla similar a la siguiente:

IP	Free_IP_block
.1	.1 - .64
.128	.128 - .254
.65	.65 - .127

Puede verse la relación entre una y otra de forma clara. En cada tabla tenemos una entrada por cada nodo de la red. También se cumple que ambas tablas deben estar actualizadas constantemente: deben reflejar la entrada o salida de cualquier nodo de la red.

Por lo tanto, ya que OLSR se encarga de conocer la topología de la red en todo momento, basta con monitorizar su tabla de encaminamiento para detectar cuándo un nodo se une o abandona la red. Es decir, la tabla de bloques de direcciones IP libres se actualizará a la par junto a la tabla de encaminamiento con el mecanismo provisto por OLSR.

Veamos un ejemplo que ilustra el funcionamiento y la actualización de los campos. Hablaremos de las estructuras relacionadas con la sincronización,

omitiendo los mensajes necesarios para realizar estos pasos. En la sección 8.2 se explica con detalle el protocolo, los mensajes necesarios, y su orden.

La siguiente tabla muestra todos los nodos de una red. Tenemos un único nodo, con 254 direcciones IP libres.

IP	Free_IP_block
.1	.1 - .254

Esta es la tabla que maneja el protocolo de SARA internamente.

Como se ve, la propia dirección del nodo está dentro del bloque de direcciones IP libres. Más adelante veremos por qué esto es necesario, y cómo se garantiza que no sea un problema.

Supongamos ahora que un nuevo nodo entra en la red. Mediante los mensajes definidos más adelante en el apartado 7.4, este nuevo nodo obtiene una dirección IP y un bloque de direcciones IP libres del nodo con la dirección IP **1.**

El bloque de direcciones libres [.1 - .127] se divide en dos sub-bloques de 127 direcciones cada uno:

[.1 - .127] y [.128 - .254]

El nodo con la dirección **.1** debe entregar uno de estos dos sub-bloques al nuevo nodo cliente, y este será el que no contiene su propia dirección (**.1**).

El nodo que entra en la red tiene por tanto asignado el rango de direcciones libres [.128 - .254], del que tomará la primera dirección para su propio uso.

En el caso de que el nuevo nodo tenga más de una interfaz de red participando en la misma red MANET, usará tantas direcciones como interfaces. Esas direcciones serán las primeras del rango. La primera de las

direcciones de sus interfaces será además su dirección principal, que identificará al nodo; cada dirección secundaria está ligada a la dirección principal del nodo.

En este momento, el nuevo nodo **.128** estará correctamente configurado y comenzará a utilizar el encaminamiento OLSR. Por lo tanto, el nodo **.1** descubrirá gracias a OLSR una nueva ruta hacia el nodo **.128**.

Esta nueva ruta añadida a la tabla de encaminamiento se monitoriza, para desencadenar que ese nuevo nodo sea añadido a la tabla de autoconfiguración:

IP	Free_IP_block
.1	.1 - .127
.128	.128 - .254

En este momento un nuevo nodo entra en la red, y de nuevo es el nodo **.1** quien le proporciona una dirección IP y un bloque de direcciones libres.

En este caso, **.1** dispone de 127 direcciones libres que debe dividir en dos sub-bloques. Como no puede crear dos sub-bloques de 63.5 direcciones cada uno, creará los sub-bloques

[.1 - .64] y [.65 - .127]

de 64 y 63 direcciones respectivamente. De nuevo, la dirección que está usando (**.1**) está contenida en el sub-bloque izquierdo, por lo que se asigna el derecho al nuevo nodo:

IP	Free_IP_block
.1	.1 - .64
.128	.128 - .254
.65	.65 - .127

Veamos ahora qué ocurre si el nodo **.128** abandona la red. Al dejar de participar en la red, las direcciones libres [.128 - .254] no serán asignadas por **.128** a nuevos nodos.

Estas direcciones no pueden desperdiciarse, por lo que alguien tiene que recogerlas. El encargado de ello es el único nodo que puede añadir esas direcciones por la derecha al bloque de direcciones IP libres que ya tiene. Ese nodo es **.65.**

IP	Free_IP_block
.1	.1 - .64
.65	.65 - .254

Recordemos que esta tabla es una estructura interna que maneja el protocolo SARA. Cada uno de los nodos de la red tiene una copia, y cada nodo realiza las actualizaciones que estamos explicando de forma local, sin intercambiar ningún mensaje.

Aquí hay que destacar que el nodo **.1** fue quien facilitó la entrada a la red al nodo **.128**. En el protocolo anterior diríamos que **.1** es el padre de **.128**. Sin embargo, hemos visto que el encargado de recoger las direcciones que **.128** deja disponibles es otro nodo diferente, determinado únicamente por el bloque de direcciones que tiene asignado.

Como vemos, el bloque de direcciones libres del nodo **.65** ha crecido, pero sigue manteniéndose como un único bloque de direcciones contiguas.

En el caso de que **.1** dejase la red, su bloque de direcciones [.1 - .64] debe ser recogido por otro nodo. En este caso, no hay ningún nodo que pueda añadir este bloque al suyo por la derecha, ya que la dirección más baja de la red (**.1**) está contenida en él.

Por lo tanto, sus direcciones tienen que ser recogidas por el nodo que pueda añadir este bloque que queda disponible al suyo por la izquierda. De nuevo es el nodo **.65** el responsable.

IP	Free_IP_block
.65	.1 - .254

Se observa una situación nueva: la dirección IP del nodo (**.65**) está contenida en mitad de su bloque de direcciones IP libres, en lugar de ser la primera de él.

Si no permitiésemos esto, el bloque no sería uno sólo, sino que tendríamos las direcciones libres

[.1 - .64] + [.66 - .254]

y ya no se cumpliría que cada nodo tiene un único bloque de direcciones libres.

En realidad esto no supone un problema a la hora de realizar la división binaria de direcciones libres. La entrada de un nodo en este momento supondría que el nodo **.65** ha de dividir su bloque de direcciones. Su propia dirección IP queda en el sub-bloque izquierdo, y puede desprenderse de las direcciones .128 - .254 sin problemas.

En este punto la tabla quedaría de la siguiente forma:

IP	Free_IP_block
.65	.1 - .127
.128	.128 - .254

Si otro nodo entra en la red, y es **.65** de nuevo quien le proporciona su dirección IP, las direcciones IP libres tendría que dividirlas en los sub-bloques:

[.1 - .64] y [.65 - .127]

De los dos bloques, uno de ellos es el que contiene la dirección IP del nodo que facilita la entrada al nuevo (la dirección IP **.65** está en el bloque [.65 - .127]).

En este caso, al contrario de lo que ha sucedido antes, se entregará al nuevo nodo el bloque con las direcciones más bajas, para evitar que la dirección IP **.65** sea usada por cualquier otro nodo; ya que en realidad no está libre.

IP	Free_IP_block
.65	.65 - .127
.128	.128 - .254
.1	.1 - .64

I. Casos especiales, problemas de sincronización

Al realizar la actualización de la tabla de autoconfiguración cada nodo de forma local, sin intercambiar mensajes que confirmen los cambios, se debe tener la certeza de que ninguna situación puede desencadenar que un nodo tenga almacenada información que no se corresponda a la situación real de la red.

Por ello, se estudiaron diferentes escenarios que podrían ser problemáticos. Existen varias situaciones conflictivas con el modelo de sincronización propuesto.

Un nodo perteneciente a la red ofrece la mitad de su bloque de direcciones IP libres a los nodos que intentan acceder, y así lo solicitan. Se debe tener en cuenta que tras ofrecer esas direcciones IP, el nodo servidor podría recibir otra solicitud diferente. En ese caso, el nodo no debería responder a esa solicitud hasta tener confirmación de que el ofrecimiento anterior ha sido aceptado, o rechazado.

Conocer si el nodo cliente al que se le han ofrecido las direcciones IP libres ha aceptado la solicitud es simple, ya que el protocolo OLSR detectará una nueva ruta hacia un nodo con la primera dirección IP del bloque ofrecido.

Para detectar si una solicitud no ha sido aceptada, el nodo servidor pone en marcha un temporizador desde el momento en que se envía el primer ofrecimiento. Si al expirar este temporizador no se tiene constancia de que el nodo cliente ha aceptado el ofrecimiento, esto significa que ha sido rechazado, ya que la oferta habrá caducado y el nodo cliente no podrá usar las direcciones ofrecidas.

Otro escenario problemático al que se debe prestar atención es la detección de entradas o salidas de nodos fuera de orden. Para explicar este escenario, desarrollamos a continuación un ejemplo práctico.

Supongamos que la siguiente tabla representa parte de una red en tres instantes de tiempo diferentes, t1, t2 y t3. Se representan tres nodos, y el bloque de direcciones que le corresponde a cada uno en los diferentes instantes. En un primer momento, el único nodo de la red representada es el nodo A. Más adelante se incorporan a esta los nodos B y luego C, siendo en ambos casos el nodo A el que les facilite la entrada.

Nombre	IP	t1	t2	t3
A	1	1 – 100	1 – 50	1 – 25
B	51		51 – 100	51 – 100
C	26			26 – 50

Si esta es parte de una red más grande, estos cambios son detectados por otro nodo perteneciente a ella, aquí no representado. En su tabla de autoconfiguración, tiene almacenada la información de que el nodo A usa la dirección IP **1** y el bloque de direcciones IP libres [1 – 100]. Mediante la monitorización de la tabla de encaminamiento del protocolo OLSR, detectará la entrada de los nodos B (**51**) y C (**26**).

Pero por problemas de interferencias, saturación de la red, o pérdida de mensajes, podría ocurrir que OLSR incluyese antes una ruta hacia el nodo C que hacia el nodo B. Desde el punto de vista de la autoconfiguración, lo que pasaría es que en el instante t2 aparece un nodo con la dirección IP **26**. Esto implica que esa dirección ha sido facilitada por A, ya que tenía asignado el bloque de direcciones [1 – 100], lo cual es incorrecto. A habría dividido su bloque de direcciones en dos sub-bloques, [1 – 50] y [51 – 100], por lo que la aparición de un nodo con la dirección IP **26** no es posible.

Nombre	IP	t1	t2	t3
A	1	1 – 100	1 – **50**	
B	-			
C	**26**		**51** – 100	

Un problema muy similar a este se da si un nodo abandona la red en un momento cercano en el que el servidor que debe recoger las direcciones que

quedan disponibles está dando entrada a un nuevo cliente. Otro nodo de la red podría detectar la nueva entra y la salida del nodo de manera que la tabla de autoconfiguración resultante no tuviese sentido.

La solución propuesta consiste en establecer temporizadores, de manera que se garantice que la entrada de un nuevo nodo en la red no esté cercana a otra entrada, o salida, que modifique los bloques de direcciones IP involucrados en esa nueva entrada.

Tras otorgar un bloque de direcciones IP libres a un nuevo nodo cliente, el servidor establece un tiempo durante el cual no atenderá más peticiones. Asimismo, el nodo que acaba de entrar en la red esperará el mismo tiempo hasta comenzar a atender peticiones de otros nodos.

Siguiendo con el ejemplo anterior, tras facilitar la entrada al nodo B, el nodo A no atendería la solicitud de C hasta que pasase un tiempo suficiente como para que cualquier nodo de la red haya podido detectar la nueva presencia de B. Por lo tanto, no existe riesgo de que la entrada del nodo C sea detectada antes que la de B por algún otro nodo.

De manera similar, cuando un nodo detecta que debe recoger las direcciones que un nodo que acaba de abandonar la red deja disponibles, espera un tiempo suficiente antes de volver a atender peticiones de clientes. Así, el resto de la red actualizará el bloque de direcciones correspondiente antes de que se divida y se entregue la mitad a algún cliente.

8.1.4. Inicialización

Hemos visto que podemos actualizar la información necesaria para sincronizar el protocolo SARA aprovechando las actualizaciones de rutas de OLSR. Pero para poder actualizar la tabla, un nodo que acaba de entrar a la red debe primero construirla.

Proponemos dos soluciones: comenzar con una tabla, o ir creándola con información añadida a las actualizaciones periódicas del protocolo de encaminamiento.

En el código para realizar las simulaciones se implementó la primera solución, ya que como veremos a continuación es más simple y eficiente al no requerir modificar el protocolo OLSR.

Paso de tabla entera en un mensaje

Cuando un nuevo nodo accede a la red, depende de la intervención de un nodo cualquiera de los que ya pertenecen a la red para obtener una dirección IP válida y un bloque de direcciones IP libres, que el nuevo nodo podrá proporcionar a su vez a futuros nodos que deseen entrar a la red.

Durante ese proceso, el nuevo nodo está configurándose para poder hacer uso de la red, pero todavía no pertenece a ésta. Por ello, se decidió que en ese mismo momento sería lógico que el nodo servidor transmitiese una copia de su tabla de autoconfiguración al nodo cliente.

De este modo, se decidió modificar el mensaje *IP_ASSINGMENT* para que contenga la tabla mencionada.

Esto quiere decir que en el momento en el que un nodo comienza a participar en la red dispone de una tabla de autoconfiguración que puede actualizar con la información aportada por OLSR.

Hay que aclarar que aunque la información sobre los bloques de direcciones libres y la topología de la red están relacionadas, en este primer momento se dispone de la tabla de autoconfiguración, pero OLSR comenzará a funcionar de aquí en adelante. La tabla de rutas se encuentra vacía.

Si OLSR detecta una ruta nueva hacia un nodo antes desconocido, se notificará además la tabla de autoconfiguración. Si ese nodo que OLSR descubre como nuevo se encuentra en la tabla que se recibió del nodo servidor, no habrá que actualizar ninguna entrada.

Se puede dar el caso de que tras recibir esa tabla de autoconfiguración con todos los nodos de la red, se produzca la salida de uno de ellos. Como se ha dicho, durante un tiempo inicial OLSR no contiene ninguna ruta, y las irá descubriendo desde ese momento. Por lo tanto, desde el punto de vista de

OLSR, el que un nodo desaparezca de la red en esos primeros instantes no implica borrar una ruta, ya que no se disponía de ninguna. En este caso la detección de esa salida se realiza mediante un temporizador.

Al recibir la tabla, se inicia un temporizador. Cuando este expira, se borrarán de la tabla de autoconfiguración a los nodos para los cuales OLSR no tenga ruta conocida. De este modo, el problema que se daría porque un nodo abandone la red sin que OLSR lo detecte queda resuelto.

Tras cubrir cómo se resolvieron la sincronización y la inicialización, se observa que el protocolo OLSR es monitorizado, pero no se modifica ni su comportamiento, ni sus mensajes. Esto implica que la solución propuesta genera una **sobrecarga nula** sobre el tráfico originado por OLSR una vez que el nuevo nodo esté configurado correctamente.

Construirla a partir de información incluida en los mensajes OLSR

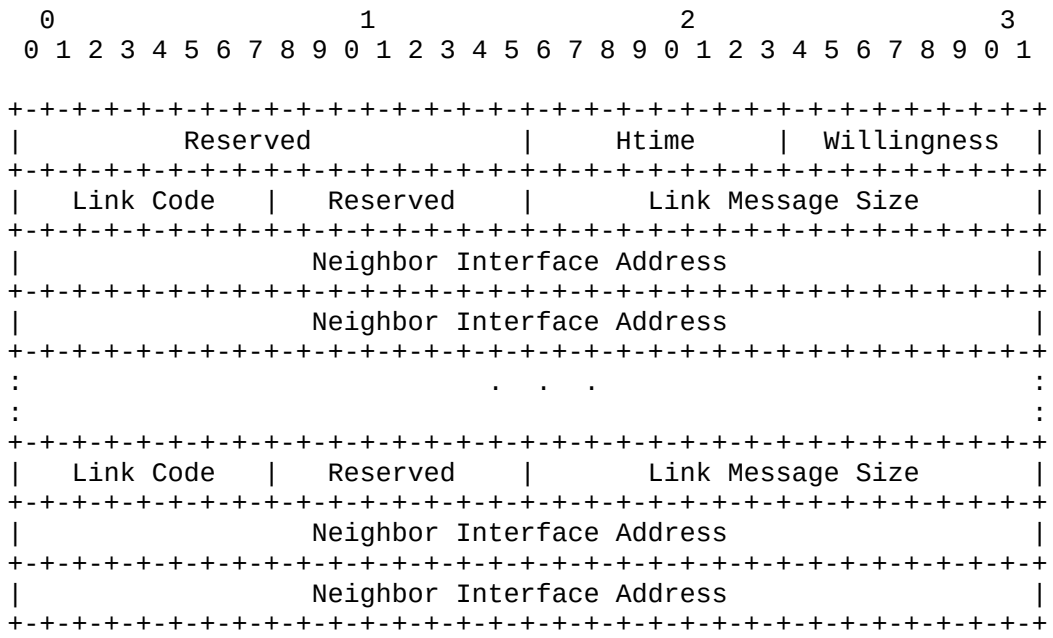
Al estudiar la forma de proporcionar a los nuevos nodos de la red una tabla de autoconfiguración, se propuso una segunda opción. Aunque no se implementó, de haber dispuesto de más tiempo hubiese sido interesante realizar las dos implementaciones y disponer de gráficas comparativas.

Esta segunda propuesta consiste básicamente en que cada nodo anunciaría su propio bloque de direcciones IP libres en los mensajes de control OLSR.

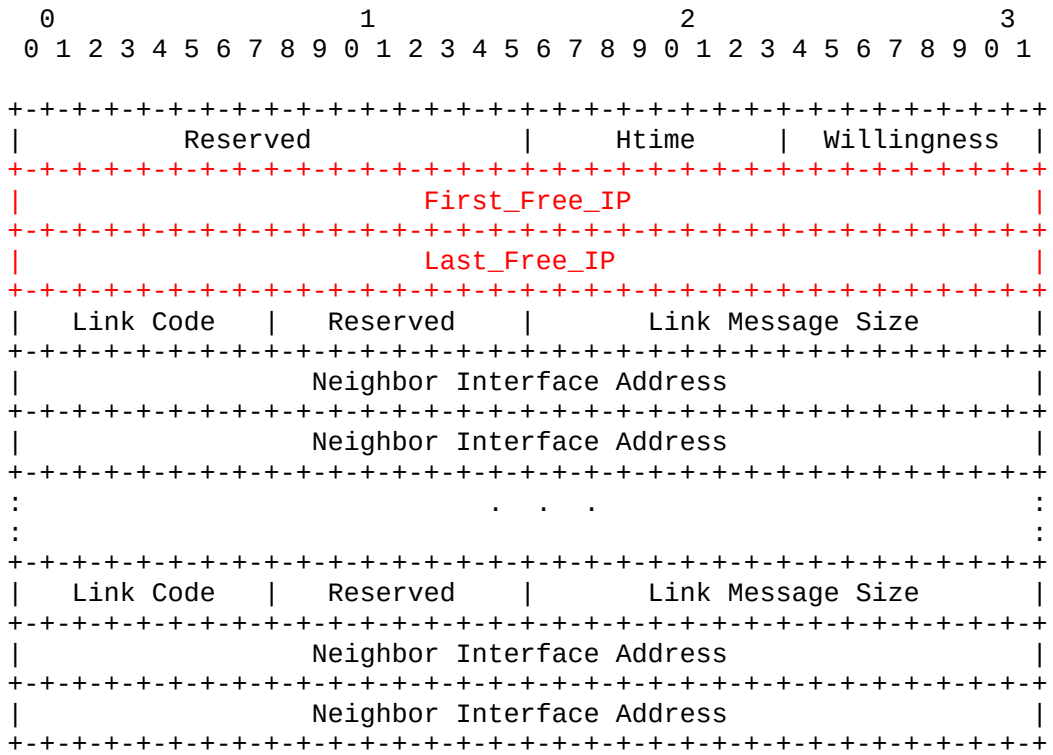
De esta forma, un nodo que entre en la red sería capaz de conocer los bloques que corresponden a cada nodo de la red al recibir las actualizaciones OLSR. Así construiría su tabla de autoconfiguración al mismo tiempo que la de encaminamiento.

Los mensajes que necesitarían nuevos formatos son los siguientes:

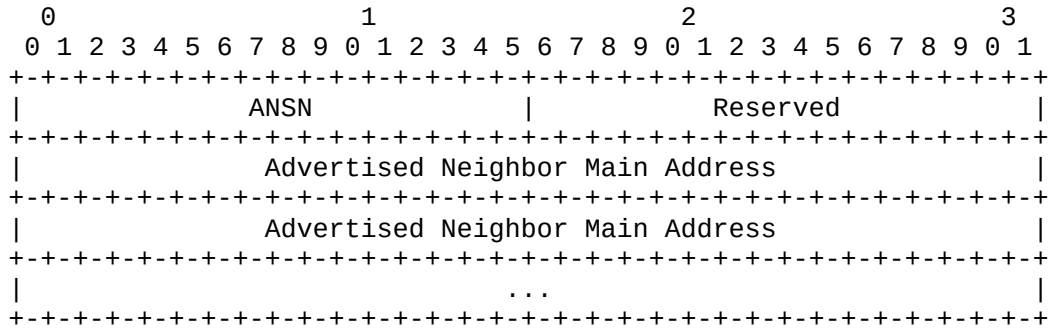
Los mensajes *HELLO*, usados para que los nodos se den a conocer a sus vecinos inmediatos, tienen los siguientes campos:



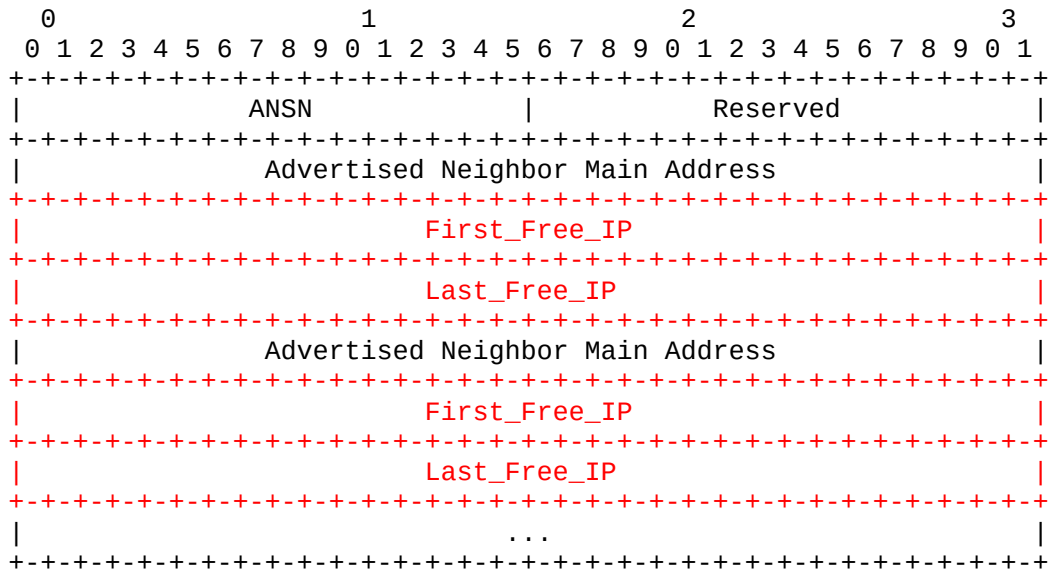
Para que el nodo que lo emite anuncie su bloque de direcciones IP libres, habría que incluir dos campos, la primera y la última dirección del bloque.



El otro tipo de mensajes de control que se utilizan para difundir la información sobre la topología de la red son los TC. Este es el formato de estos mensajes:



Para incluir el bloque de direcciones libres de cada nodo, se necesitarían dos campos más por cada nodo anunciado. El formato resultante sería el siguiente:



Modificar los mensajes del protocolo OLSR implicaría una mayor dificultad, y una dependencia mayor entre el protocolo de autoconfiguración y el protocolo de encaminamiento.

Aparte de esto, se puede ver que se generaría una sobrecarga con respecto al tráfico normal de OLSR. Por otra parte, esta sobrecarga está definida desde el primer momento: será de 64 bytes en los mensajes *HELLO*, y 64 bytes por cada vecino contenido en un mensaje TC. Conociendo este dato, se podría estudiar en qué tipos de redes la sobrecarga no sería un factor problemático, y si las ventajas superarían a las desventajas.

Realizar estos cambios tendría otras implicaciones. Por ejemplo, al difundir el bloque de direcciones libres de cada nodo, los nodos de la red no actualizarían de forma local más que su propio bloque de direcciones libres.

En lugar de suponer que al abandonar la red un nodo A, otro nodo B recogerá las direcciones que A deja disponibles, el resto de la red podría esperar a que B anuncie que de hecho las ha recogido. Por ello, podría incluirse algún método de comprobación, como podría ser el envío de varios mensajes *ping* desde B hasta A, ya que el único nodo que debe asegurarse de la salida de A es B, y no toda la red.

Otra idea que podría desarrollarse sobre la base de difundir los bloques de direcciones libres de cada nodo es la posibilidad de que un nodo que quiera entrar a la red sea capaz de conocer el estado de ésta de antemano, escuchando los mensajes OLSR. Esto podría facilitar otros mecanismos de autoconfiguración más complejos, como podría ser el que el nodo que quiera entrar a la red eligiese al nodo de la red que más le interese, según sus criterios, como su servidor para que le facilite la entrada.

8.2. Especificación del nuevo protocolo

8.2.1. Introducción

En esta sección se expone una especificación del protocolo propuesto en detalle y de manera rigurosa.

SARA es un protocolo de autoconfiguración con seguridad que gestiona la entrada y salida de nodos en redes MANET. Incluye un módulo, explicado en el capítulo 9, destinado a proporcionar seguridad a las redes MANET.

El protocolo hace que los nodos de una red MANET colaboren entre sí para gestionar la asignación de direcciones IP únicas y correctas de forma distribuida. Todos los nodos de la red tienen el mismo papel, no existe un tipo de nodo especial que centralice la gestión de la misma.

Los nodos cuentan con un sistema de sincronización que se apoya en el protocolo de encaminamiento OLSR. Gracias a este mecanismo, la sincronización se lleva a cabo de forma pasiva, monitorizando el protocolo de encaminamiento mencionado, por lo que se genera una sobrecarga nula en el tráfico de la red con respecto al generado por el protocolo OLSR.

Al ser todos los nodos responsables de gestionar la entrada de cualquier otro nuevo nodo a la red, esta operación puede realizarse rápidamente. Un nodo que desea entrar a una red intenta contactar con cualquier nodo ya perteneciente a ella, y podrá recibir varias respuestas de varios nodos. Esto hace que las posibilidades de entrar a formar parte de la red con éxito sean altas, debido a la alta disponibilidad y a la redundancia que supone la gestión distribuida.

La estructura de esta sección, que contiene la especificación del protocolo SARA, es la siguiente: comienza con las estructuras de datos usadas, continúa con una explicación de los mensajes que se intercambian entre los nodos para la entrada y salida de éstos, y a continuación se detalla cómo se lleva a cabo la sincronización en el protocolo.

A continuación se hablará de cómo se solucionan las posibles pérdidas de mensajes en la red, usando los temporizadores adecuados y realizando determinadas acciones cuando estos expiran para reestablecer el proceso de autoconfiguración.

Para acabar, se explicará el formato de los mensajes y los diagramas de estados para cada modo de funcionamiento que puede asumir un nodo.

8.2.2. Estructuras de datos

Las estructuras de datos de este protocolo pueden clasificarse en las propias que maneja el mecanismo de autoconfiguración y las pertenecientes al protocolo de encaminamiento OLSR.

OLSR almacena internamente, entre otros, una tabla de rutas que es actualizada periódicamente. Esta tabla contiene información sobre la ruta a cada nodo, almacenada en los siguientes campos:

- **R_dest_addr**: Dirección IP del nodo destino.
- **R_next_addr**: Dirección IP del siguiente salto en la ruta.
- **R_dist**: Distancia al nodo destino.
- **R_iface_addr:** Dirección IP de la interfaz de salida al nodo destino.

Las estructuras necesarias para el mecanismo de la autoconfiguración son:

- Direcciones IP de las interfaces del nodo
- Máscara de red
- Free_IP_Blocks: tabla de bloques libres de cada nodo de la red. Tendrá la siguiente forma:

IP	*Free_IP_block*
.1	.1 - .64
.128	.128 - .254
.65	.65 - .127

8.2.3. Entrada y salida de los nodos

El protocolo usa un número concreto de mensajes para cada operación. Todas las operaciones están definidas buscando un óptimo funcionamiento y una baja latencia.

En esta sección se habla de cómo se establece la comunicación entre los nodos y los mensajes transmitidos durante la salida y la entrada de nodos en la red.

I. Entrada de nodos

La entrada de un nodo a la red implica la necesidad de encontrar un nodo que actúe como servidor. Una vez encontrado, éste le facilita la entrada proporcionándole un bloque de direcciones IP y una tabla *Free_IP_Blocks* que representa el estado de los nodos de toda la red.

Hasta que el nodo no tenga asignada una dirección IP, su comunicación con los nodos que podrán actuar como servidores será a través de la capa MAC.

Este mecanismo de configuración utiliza 4 tipos de mensajes en la mayoría de los casos. Si no hay nodos en su rango con direcciones IP libres, se usarán 6 tipos de mensajes en total.

En la figura 8.1 se muestra el esquema de intercambio de mensajes que se explica a continuación:

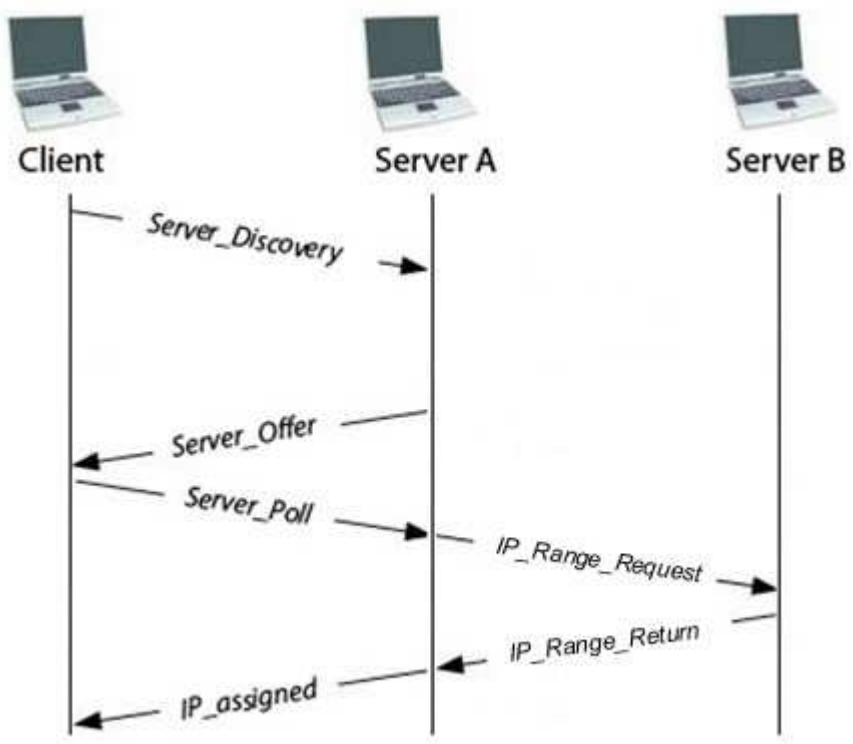


Fig. 8.1: Paso de mensajes en el proceso de entrada de un nodo a la red

El proceso de intercambio se realiza de la siguiente forma:

- **SERVER_DISCOVERY:**

  El nodo cliente que desea entrar en una red inicia el proceso con un mensaje de este tipo. Se transmite por la capa MAC, con la dirección de difusión como destino. En el mensaje se indican el número de direcciones IP que se necesitan (igual al número de interfaces).

  Si el nodo dispone de más de una interfaz de red, el mensaje se transmite por todas ellas, usando el campo ID para que las diferentes interfaces no se confundan con varios nodos.

- **SERVER_OFFER:**

  Los nodos de la red que reciben el mensaje SERVER_DISCOVERY responden con este mensaje, también usando la capa MAC, en el que ofrecen un número de direcciones IP. El número de direcciones ofrecidas es la mitad del rango disponible.

  El mensaje SERVER_DISCOVERY incluye un campo *Count* que indica cuántos intentos se han realizado por parte del cliente. Dependiendo de su valor, los nodos servidores se comportarán del siguiente modo:

  - **Count = 1**: El nodo servidor responderá con un SERVER_OFFER si dispone de direcciones suficientes y los campos R (Ready) y L (Local) tendrán el valor 1 (puede asignar las direcciones ofrecidas en este momento, y son direcciones del bloque del propio nodo).

  - **Count = 2**: El nodo servidor responderá con un SERVER_OFFER si los campos pueden tomar el valor R = 1 y L = 1. En caso de no ser posible, responderá también si se da la situación de que tiene suficientes direcciones y R = 0, L = 1 (el servidor no puede asignar las direcciones en este momento, pero dispone de ellas).

  - **Count > 2:** Si el nodo dispone de direcciones y por su estado es capaz, enviará un SERVER_OFFER con R = 1, L = 1. Si puede, lo enviará con R = 0, L = 1. Y por último, si no dispone de suficientes direcciones libres, enviará el mensaje con los campos R = 1, L = 0

(disponibilidad inmediata de las direcciones, pero las direcciones ofrecidas son de otro nodo de la red).

- **SERVER_POLL.**

  Tras un tiempo de escucha, el nodo cliente habrá recibido varios mensajes SERVER_OFFER. De no ser así, volvería a intentarlo (ver diagramas de estados en la sección 8.2.7).

  Ordenará los mensajes recibidos según los siguientes criterios:

  - Se descartan los servidores que no estén disponibles, es decir, con R = 0. Los SERVER_OFFER con R = 0 no se usan para poder responder con un SERVER_POLL, pero tienen la función de informar al cliente de que existe un nodo servidor de una red aunque en ese momento no pueda facilitarle el acceso.

  - Se dan prioridad a las direcciones locales: preferirá los mensajes con el campo L = 1.

  - Por último, se ordenarán por número de direcciones ofrecidas, de mayor a menor.

  Según ese orden de preferencia, enviará al primer servidor un mensaje SERVER_POLL (de nuevo, por la capa MAC) para indicarle que le ha elegido para que le asigne un bloque de direcciones IP libres.

- **IP_RANGE_REQUEST**

  Si las direcciones ofrecidas por el nodo servidor no eran propias, sino de un tercer nodo de la red, con este mensaje se le solicitan formalmente a ese nodo. Al ser una comunicación entre dos nodos ya configurados correctamente, se realiza en la capa IP.

- **IP_RANGE_RETURN**

Ese tercer nodo de la red autoriza al nodo que envió el mensaje IP_RANGE_REQUEST a asignar al nodo cliente el bloque de direcciones indicado en este mensaje. También es un mensaje enviado por IP.

- **IP_ASSIGNED**

  Tras recibir el SERVER_POLL, si las direcciones ofrecidas eran del propio nodo servidor, o tras el mensaje IP_RANGE_RETURN en caso de que se haya tenido que pedir las direcciones a un tercer nodo, el nodo servidor envía este mensaje al cliente. Este mensaje es transmitido por la capa MAC.

  En él se indica el bloque de direcciones libres que se le asignan al cliente, y la tabla *Free_IP_Blocks* que representa el estado de la red. La tabla que se transmite en este mensaje no refleja la entrada del nodo cliente.

Tras este intercambio de mensajes, el nodo cliente escoge como su dirección IP la primera del bloque que se le ha asignado. En caso de tener más de una interfaz de red, usará las primeras del bloque en orden, y será la primera de todas la que use como dirección principal que identifica al nodo.

II. Salida de nodos

El mecanismo de salida de nodos no requiere del intercambio de ningún mensaje. El nodo que quiera abandonar la red no tiene que avisar a ningún otro nodo de su salida, evitando la sobrecarga que estos mensajes producen.

El resto de nodos de la red, se darán cuenta de la salida del nodo mediante las actualizaciones periódicas de rutas que el protocolo OLSR realiza cada cierto tiempo. Observarán que se ha perdido la ruta hacia ese nodo, y por lo tanto lo eliminarán de su tabla *Free_IP_Blocks*, añadiendo su bloque de direcciones libres al nodo que corresponda según se ha explicado en el punto anterior.

8.2.4. Sincronización

La sincronización se lleva a cabo monitorizando la tabla de rutas del protocolo de encaminamiento OLSR.

La entrada o salida de un nodo a la red se detecta cuando OLSR añade una nueva ruta a su tabla de encaminamiento, o borra una de las existentes. Al detectar la entrada o la salida de un nodo de la red, se actualiza localmente, y sin intercambiar ningún mensaje, la tabla *Free_IP_Blocks*.

Para ello, se siguen las siguientes reglas:

- El responsable de recuperar las direcciones IP que un nodo que abandona la red deja disponibles es aquel que puede unir por la derecha ese bloque libre con el suyo.

  Esto no será posible cuando el bloque que se debe recoger contiene la dirección más baja de la red. En ese caso, el nodo que recoge el bloque es aquel que puede añadirlo al suyo por la izquierda.

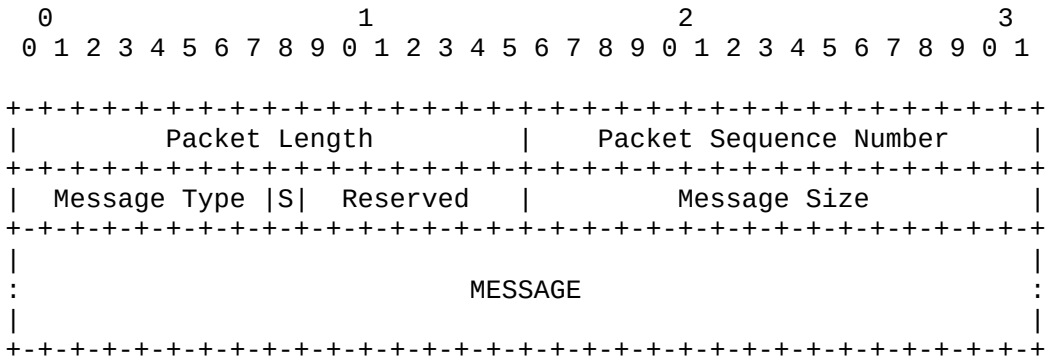
- Al dividir las direcciones libres en dos bloques para entregar uno de ellos a un nuevo nodo que entra a la red, el nodo que hace de servidor entrega al cliente el sub-bloque que no contiene su propia dirección IP.

Cuando se detecta la salida de un nodo, se debe eliminar su entrada, y actualizar la del nodo a quien corresponden las direcciones IP que han quedado disponibles.

Al detectar la entrada de un nuevo nodo, se creará una nueva entrada en la tabla para él, y se actualizará el bloque de direcciones libres del nodo que le facilitó su dirección IP. Para saber quién fue ese nodo que actuó como servidor, basta con buscar qué nodo tiene la dirección IP del nuevo nodo en su bloque de direcciones libres.

8.2.5. Formato de los mensajes

Todos los mensajes enviados en el protocolo van empaquetados con el siguiente formato:



Estos mensajes serán encapsulados a su vez con las cabeceras correspondientes a la capa MAC o TCP/IP, dependiendo del tipo de mensaje que se incluya en el campo *MESSAGE.*

Los campos significan lo siguiente:

La primera fila es la cabecera del paquete con los campos:

- **Packet Length:** Longitud del paquete, incluyendo la cabecera (2 bytes).
- **Packet Sequence Number:** Número de secuencia (2 bytes). En cada mensaje diferente que envía un nodo este campo se incrementa en uno. Sirve para poder detectar paquetes duplicados.

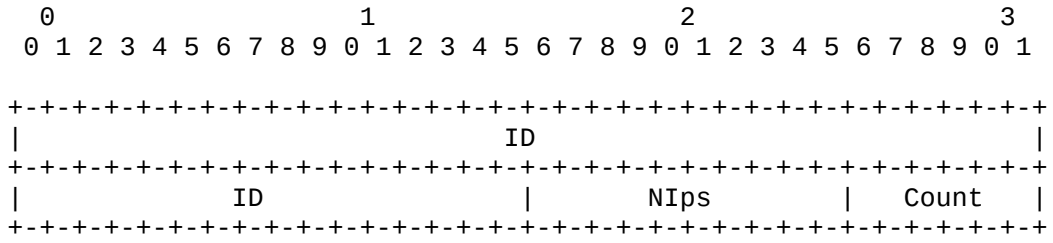
La segunda fila constituye la cabecera para cada uno de los mensajes del protocolo:

- **Message Type**: Tipo de Mensaje (1 byte).
- **S (*Security*):** Indica si están presenten los campos relativos a la seguridad. (1 bit).
- **Reserved**: Reservado para futuras ampliaciones de funcionalidad (7 bits).

- **Message Size:** Tamaño del Mensaje (16 bits).

A continuación se muestra el formato de cada tipo de mensaje, contenidos en el campo *Message*. Se marcan en color azul los campos referidos a la seguridad y que no participan en el proceso de autoconfiguración. Estos campos estarán presentes sólo si el campo S de la cabecera tiene el valor 1**.**

I. SERVER_DISCOVERY.

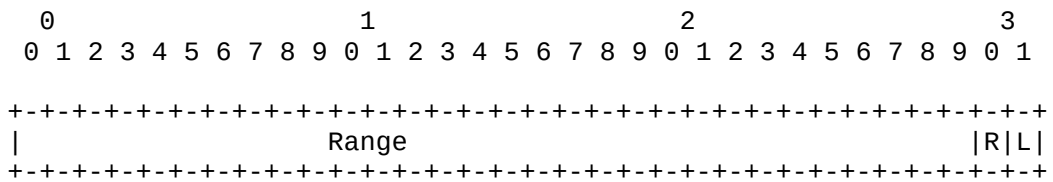


- **ID:** Identificador del nodo (6 bytes). El nodo debe elegir, si tiene más de una, la dirección MAC de una de sus interfaces. Este campo identificativo tiene el mismo valor para cualquier mensaje SERVER_DISCOVERY y SERVER_POLL emitido por el nodo aunque se haga desde distintas interfaces.

- **NIps:** Cantidad de direcciones IP solicitadas por el nodo. Será igual al número de interfaces del nodo cliente (1 byte).

- **Count:** Número de veces que se ha intentado la petición SERVER_DISCOVERY (1 byte).

II. SERVER_OFFER

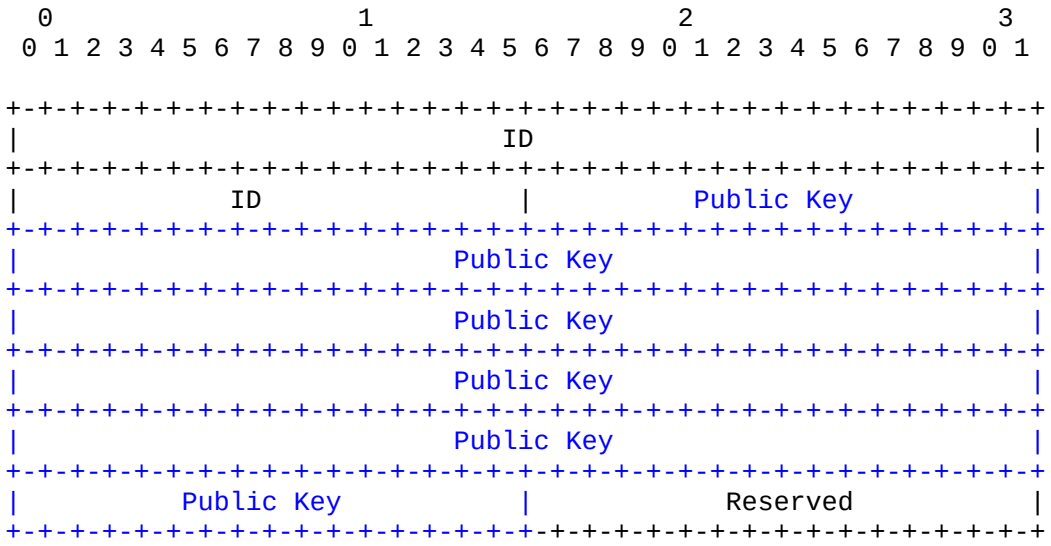


- **Range:** Número de direcciones IP que se ofrecen (30 bits).

- **R (*Ready*):** Indica si el nodo que ofrece las direcciones IP está listo para asignarlas de inmediato, o si las ofrece para comunicar su existencia aunque en este momento no esté en disposición de asignarlas (1 bit).

- **L (*Local*):** Indica si el rango ofrecido es del nodo emisor, o por el contrario se pedirá a su vez a un tercer nodo de la red (1 bit).

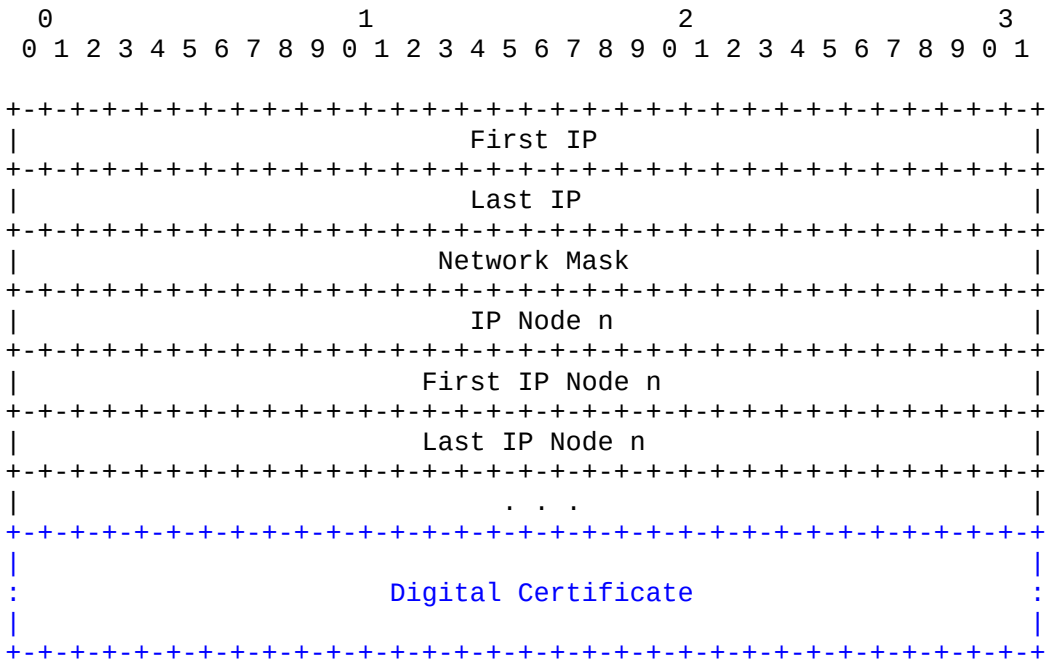
III. SERVER_POLL



- **ID:** Identificador del nodo (6 bytes) Es el mismo identificador que el elegido en el mensaje SERVER_DISCOVERY.

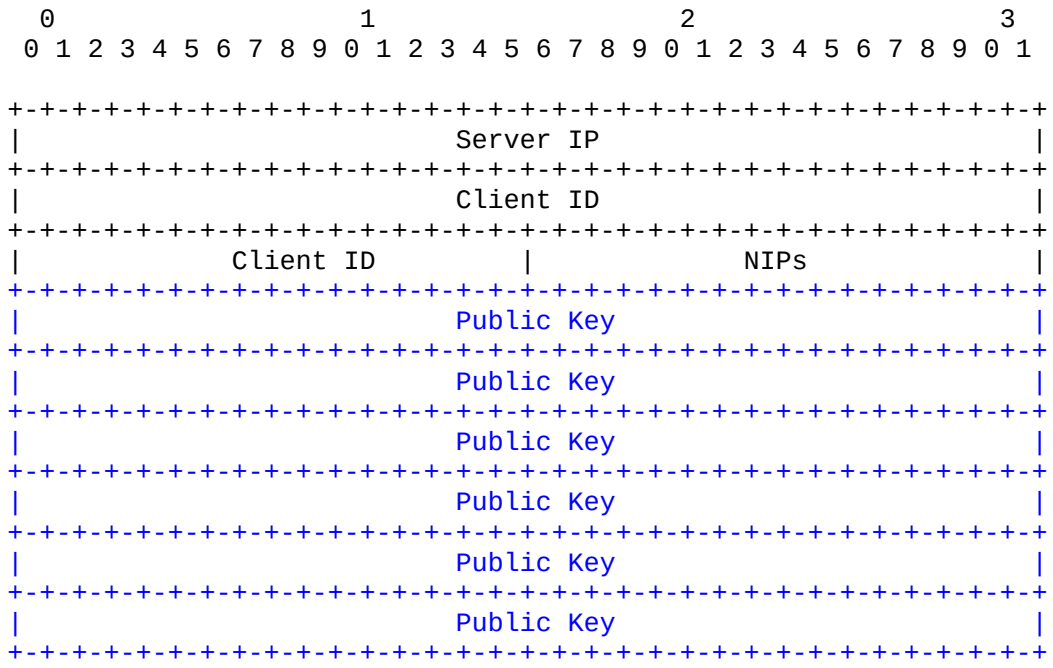
- **Public Key:** Presente sólo si S = 1 en la cabecera. Clave pública que generará el nodo cliente (20 bytes).

IV. IP_ASSIGNED



- **First IP:** Dirección IP inicial del bloque de direcciones libres (4 bytes).
- **Last IP:** Dirección IP final del bloque de direcciones libres (4 bytes).
- **Network Mask:** Máscara de red (4 bytes).
- **IP Node n, First IP Node n, Last IP Node n:** Representa una entrada de la tabla de bloques libres de todos los nodos de la red. Cada nodo está representado por estos 3 campos de 4 bytes, siendo cada uno: la dirección IP del nodo, la dirección IP inicial y la final de su bloque de direcciones libres, respectivamente.
- **Digital Certificate**: Presente sólo si S = 1 en la cabecera. Certificado digital (80 bytes). Se explican los campos que lo componen en la sección 7.5.

V. IP_RANGE_REQUEST



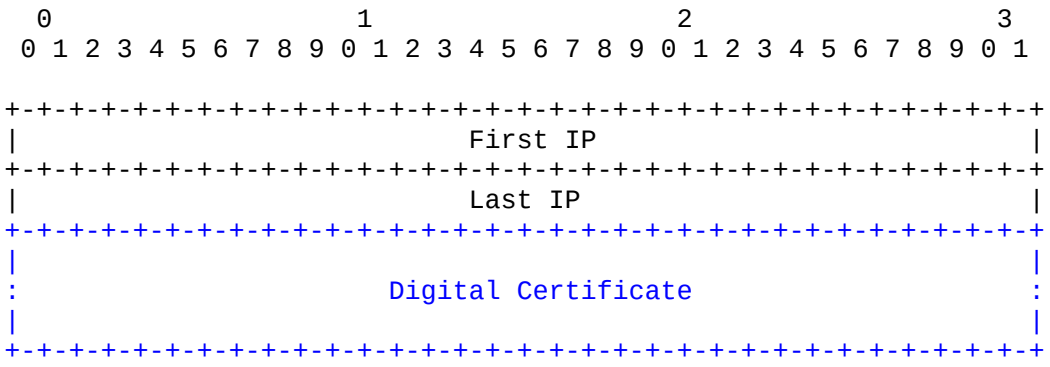
- **Server IP:** Dirección del servidor que solicita el rango para cliente (4 bytes). Puede ser diferente de la dirección IP del emisor del mensaje, al tratarse de redes multi-salto.

- **Client ID:** Identificador del nodo cliente. Tiene el mismo valor que el campo ID de los mensajes SERVER_DISCOVERY y SERVER_POLL (6 bytes).

- **NIPs:** Número de direcciones IP solicitadas (2 bytes).

- **Public Key:** Presente sólo si S = 1 en la cabecera. Clave pública del cliente. El nodo servidor emisor la ha recibido en el mensaje SERVER_POLL (20 bytes).

VI. IP_RANGE_RETURN



- **First IP:** Dirección IP inicial del bloque de direcciones libres (4 bytes).
- **Last IP:** Dirección IP final del bloque de direcciones libres (4 bytes).
- **Digital Certificate:** Presente sólo si S = 1 en la cabecera. Certificado digital (80 bytes). Se explica en la sección 9.3.

8.2.6. Temporizadores

El carácter inalámbrico y móvil de las redes MANET provoca que en este tipo de redes se den a menudo situaciones en las que se pierden mensajes, o tardan en llegar a su destino más que el tiempo estimado. Por ello, exponemos a continuación una serie de temporizadores usados para resolver situaciones de este tipo:

- **SERVER_DISCOVERY_TIMER**: Tras enviar el mensaje SERVER_DISCOVERY, el nodo cliente iniciará este temporizador al pasar al estado WAITING_REPLY. Durante este tiempo el nodo está a la espera de mensajes SERVER_OFFER de los posibles nodos cercanos pertenecientes a una red. Cuanto más largo sea el temporizador mayor tiempo se dedicará a recibir mensajes de este tipo, por lo que habrá más para procesar y, por lo tanto, será más fácil obtener un bloque de direcciones. Pero también implica el aumento de la latencia al obtener una dirección IP.

Si este temporizador expira y el nodo cliente no ha recibido ningún SERVER_OFFER, bien por pérdidas de mensajes, o porque no hay ningún nodo servidor, volverá a enviar un nuevo SERVER_DISCOVERY. Esta acción se repetirá un número máximo de veces (SDISCOVERY_MAX_RETRY) y, si sigue sin recibir mensajes, iniciará su propia red.

- **SERVER_OFFER_TIMER:** Tras el mensaje SERVER_OFFER, el nodo pasa al estado WAITING_POLL, e inicia este temporizador. Al expirar, el estado cambiará a IDLE.

- **SERVER_POLL_TIMER:** Una vez enviado el mensaje SERVER_POLL, el nodo cliente esperará al mensaje IP_ASSIGNED el tiempo que determine este temporizador. Si este mensaje no llega, se retransmitirá el SERVER_POLL hasta un número máximo de intentos, definido como SPOLL_MAX_RETRY. Si se supera el número máximo de intentos, comenzará el proceso de configuración de nuevo.

- **IP_RANGE_REQUEST_TIMER:** El nodo servidor que manda un mensaje IP_RANGE_REQUEST a otro nodo de la red inicia este temporizador en ese momento. Al igual que con el SERVER_POLL_TIMER, si expira este temporizador se reenviará el mensaje IP_RANGE_REQUEST hasta un número máximo de intentos, RREQUEST_MAX_RETRY.

- **ACCEPTED_OFFER_TIMER:** Este temporizador se activa tras enviar un mensaje IP_ASSIGNED o un mensaje IP_RANGE_RETURN. Durante este tiempo, el nodo servidor no puede responder a solicitudes de tipo SERVER_POLL o IP_RANGE_REQUEST. Esta restricción se levantará al expirar el temporizador (la oferta caducó sin ser aceptada), o al detectar que un nodo con la primera dirección IP de las ofrecidas ha entrado en la red (el bloque de direcciones ofrecido fue aceptado).

Hay que tener en cuenta que aunque no pueda asignar direcciones IP, el nodo servidor seguirá respondiendo a peticiones SERVER_DISCOVERY dando el valor 0 al campo R (READY) en el mensaje SERVER_OFFER. De

este modo, se informa al nodo cliente de la existencia del servidor, aunque no sea capaz de asignar direcciones IP inmediatamente.

- **NODE_DOWN_TIMER:** Cuando OLSR borra la ruta hacia un nodo, no se elimina inmediatamente de la tabla Free_IP_Blocks. En su lugar, se inicia este temporizador. Si antes de que el temporizador expire se vuelve a descubrir una ruta hacia el nodo, eso quiere decir que desapareció momentáneamente, pero no abandonó la red. Por lo tanto, se cancela la eliminación de la tabla Free_IP_Blocks. En el caso de que el temporizador expire y no se haya recuperado una ruta, se da al nodo por perdido y se elimina su entrada de la tabla Free_IP_Blocks, actualizando las que correspondan.

- **INIT_TABLE_TIMER:** Al recibir la tabla de autoconfiguración Free_IP_Blocks en el mensaje IP_ASSIGNED, el nodo cliente activa este temporizador. Durante ese tiempo, la tabla contiene nodos para los que OLSR aún no tiene una ruta conocida. Al expirar el temporizador, se comprueba qué nodos de la tabla Free_IP_Blocks no tienen entrada en la tabla de rutas de OLSR: esos nodos se eliminan (actualizando las entradas que correspondan), ya que son nodos que pertenecían a la red al recibir la tabla, y la han abandonado antes de que OLSR supiese de su existencia.

- **INIT_ASSIGN_TIMER**: Este temporizador lo usan tanto el nodo cliente como el servidor cuando han recibido o asignado, respectivamente, un bloque de direcciones IP. Es decir, el servidor lo inicializa al comprobar la entrada de un nodo con la primera dirección IP del bloque ofrecido en el mensaje IP_ASSIGNED, y el cliente lo inicia tras recibir el mensaje IP_ASSIGNED y configurar su dirección.

  Así se da tiempo a que toda la red pueda actualizar su tabla Free_IP_Block antes de que se produzcan más cambios. Durante ese tiempo, ignorarán mensajes SERVER_POLL o IP_RANGE_REQUEST, aunque responderán a los SERVER_DISCOVERY.

- **NODE_DOWN_ASSIGN_TIMER:** Cuando un nodo ya configurado detecta la salida de otro, y comprueba que le corresponde recoger las direcciones

IP que quedan libres, pone en marcha este temporizador. Más concretamente, el temporizador se activará cuando se detecte la eliminación de la tabla de encaminamiento OLSR, es decir, se activará al mismo tiempo que el temporizador NODE_DOWN_TIMER.

Hasta que no expire, el nodo ignorará las peticiones SERVER_POLL e IP_RANGE_REQ. De ese modo, se dará un margen de tiempo para asegurarse de que todos los nodos en la red detectan la salida mencionada y actualizan su tabla Free_IP_Blocks, antes de asignárselas a algún otro nuevo nodo. Por lo tanto, la duración de este temporizador debe ser mayor que la del NODE_DOWN_TIMER para asegurar que el resto de nodos de la red no sólo han detectado la eliminación de una ruta sino que lo han eliminado de la tabla Free_IP_Blocks.

El nodo seguirá respondiendo a los mensajes SERVER_DISCOVERY fijando el valor 0 en el campo R del mensaje SERVER_OFFER.

- **SLEEP_TIMER**: Temporizador usado por un nodo cliente cuando detecta nodos cercanos pertenecientes a alguna red, pero que no están en disposición de asignar direcciones IP en este momento. De este modo, da un margen de tiempo para permitir que acaben los procesos que les impiden asignar direcciones.

8.2.7. Diagramas de estado

Dependiendo de si están en proceso de entrar en la red, o si ya pertenecen a esta, se distinguen dos tipos de nodos: cliente y servidor. En los siguientes apartados se muestran y explican los diagramas de estados que rigen el comportamiento de ambos tipos de nodos. El estado en el que se encuentra un nodo cambia al producirse envíos o recepciones de mensajes, o cuando determinados temporizadores expiran.

I. Nodo servidor

Llamamos nodo servidor a todos los nodos de la red que están correctamente configurados, es decir, que poseen una dirección IP válida con

la que comunicarse con el resto de nodos, y un bloque de direcciones IP libres. Con este bloque de direcciones libres facilitan el acceso a los nuevos nodos, que llamaremos clientes.

El diagrama de estados es el siguiente:

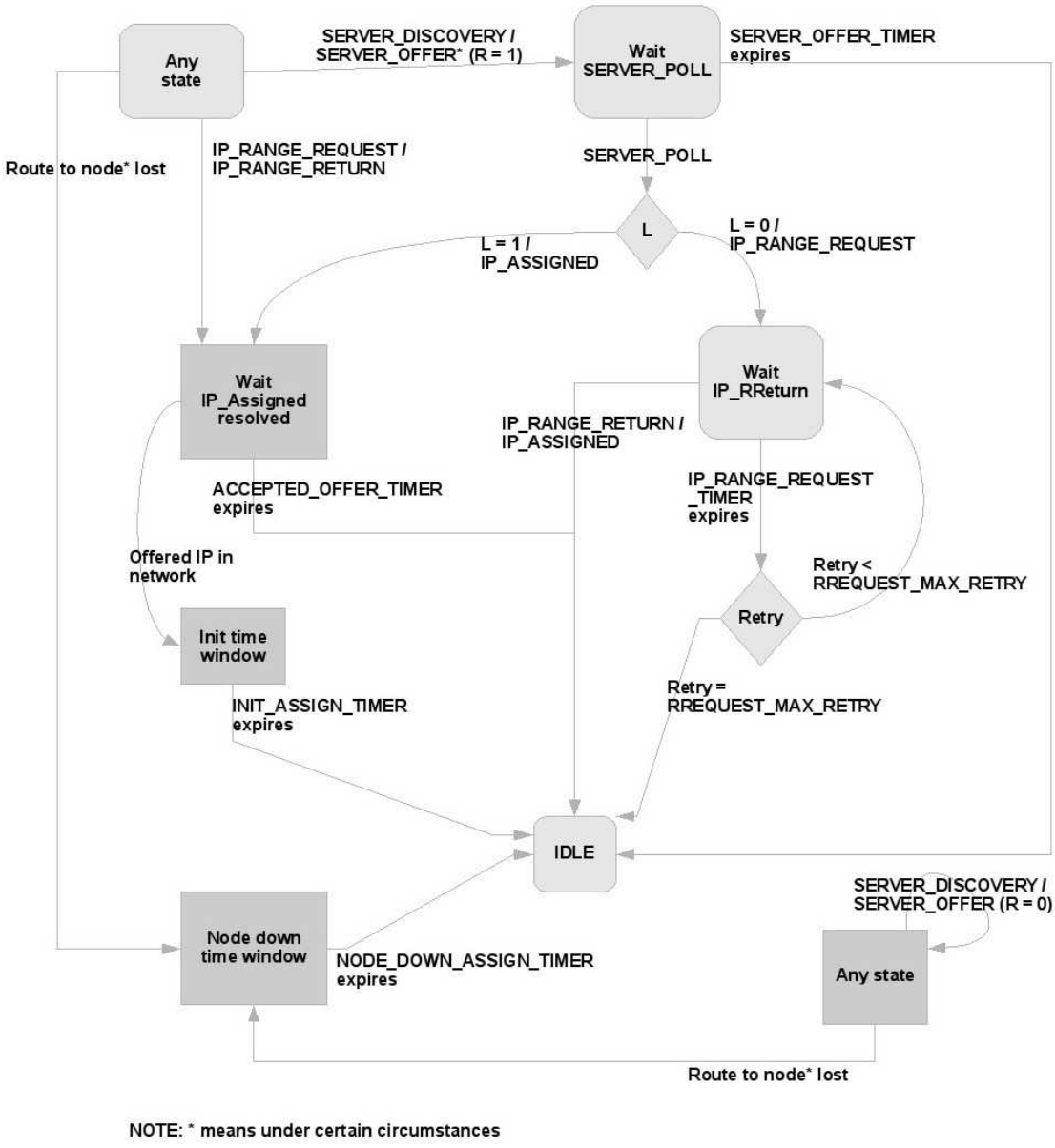


Figura 8.2 – Diagrama de estados nodo servidor.

Como puede verse, existen dos tipos de estados: los representados con rectángulos redondeados y claros, y los que se encierran en rectángulos con

esquinas y algo más oscuros. Los llamaremos estados del tipo *ready* o *not_ready*, respectivamente.

Desde cualquier estado, el nodo servidor está a la espera de mensajes SERVER_DISCOVERY. Responderá siempre con un mensaje SERVER_OFFER, pero dependiendo de si el estado actual del nodo es de tipo *ready* o *not_ready*, responderá dando al campo R (READY) el valor 1 ó 0. De este modo, un nodo siempre anuncia su presencia, aunque en ese preciso momento no sea capaz de asignar direcciones IP a un cliente. Esto se indica en el diagrama de estados con los estados *Any state*.

Any state (ready)

Desde cualquier estado de tipo *ready*, el nodo servidor responderá a un mensaje SERVER_DISOCVERY con uno de tipo SERVER_OFFER (campo R con valor 1). Eso hará que el servidor pase al estado Wait SERVER_POLL.

Un nodo puede responder al mismo tiempo peticiones SERVER_DISCOVERY de nodos diferentes, y estar a la espera de cualquiera de los correspondientes SERVER_POLL.

También se responderán a mensajes de tipo IP_RANGE_REQUEST con uno del tipo IP_RANGE_RETURN. Esto significa que si el nodo se encuentra en cualquier estado *ready*, dará la mitad de su bloque de direcciones libres a cualquier otro nodo de la red sin direcciones propias y que necesita facilitar la entrada a un cliente.

Any state (not_ready)

Mientras el nodo se encuentre en un estado de tipo not_ready, responderá a los mensajes SERVER_DISCOVERY con un mensaje SERVER_OFFER dando el valor 0 al campo R.

Wait SERVER_POLL

En este estado el nodo espera un tiempo determinado por el temporizador SERVER_OFFER_TIMER la recepción de un mensaje SEVER_POLL. Este

mensaje indica que el cliente a quien envió el SEVER_OFFER le ha elegido como su servidor para el proceso de autoconfiguración.

Tras la recepción del mensaje, el nodo enviará un mensaje IP_ASSIGNED al cliente en caso de tener direcciones disponibles localmente (el campo L del mensaje SERVER_OFFER tenía valor 1); y pasará al estado Wait IP_Assigned resolved. Si las direcciones que ofreció no eran locales, deberá pedirlas a un nodo de la red con el mensaje IP_RANGE_REQUEST; y pasará a estar en el estado Wait IP_RReturn.

En caso de que no se reciba ningún mensaje SERVER_OFFER antes de que el temporizador expire, el nodo pasará a estar en el estado IDLE.

Wait IP_Assigned resolved

En este estado, de tipo *not_ready*, el nodo está esperando a conocer si el nodo cliente recibió correctamente el bloque de direcciones ofrecido mediante un mensaje IP_ASSIGNED, o a través de un mensaje IP_RANGE_RETURN y un intermediario. El resultado puede ser que el cliente se haya configurado correctamente, o que no haya recibido el bloque de direcciones.

Esto último puede producirse por varios motivos, como pueden ser problemas de interferencias en la recepción del mensaje, moviendo del nodo cliente fuera del rango de cobertura, etc.

Si aparece un nuevo nodo en la red usando la primera dirección IP del bloque ofrecido en el mensaje IP_ASSIGNED o IP_RANGE_RETURN, significa que el nodo cliente terminó de configurarse. En ese momento el nodo servidor cambia su estado a Init time window.

Si el nodo no fue capaz de terminar el proceso de autoconfiguración, el temporizador ACCEPTED_OFFER_TIMER expirará. El nodo pasa en ese caso al estado IDLE.

Init time window

Este estado sirve para dar un margen de tiempo que permita que todos los nodos de la red sean capaces de detectar la entrada del cliente recientemente configurado, antes de volver a dividir el propio bloque de direcciones IP libres.

Si no se diese este margen, y se atendiesen peticiones nuevas de inmediato, podrían darse problemas de sincronización si otros nodos detectasen las nuevas incorporaciones a la red en un orden incorrecto.

Wait IP_RReturn

En este estado el nodo está pendiente de recibir un mensaje IP_RANGE_RETURN.

Cuando reciba ese mensaje, en el que un nodo de la red le indica un bloque que puede ofrecer al cliente en espera, enviará al cliente un mensaje IP_ASSIGNED. Tras esto, el nodo habrá terminado su función como servidor, y pasará al estado IDLE.

Si no se recibe el mensaje IP_RANGE_RETURN antes de que el temporizador IP_RANGE_REQUEST_TIMER expire, se volverá a intentar la petición enviando de nuevo un mensaje IP_RANGE_REQUEST un número máximo de veces RREQUEST_MAX_RETRY. Estos sucesivos intentos se envían en cada ocasión a un nodo diferente. Si se sobrepasa el límite de intentos, entonces el nodo desistirá y cambiará su estado a IDLE.

IDLE

Este es el estado de reposo, o en el que el nodo está ocioso. Cuando se encuentra en este estado, el nodo no está realizando ninguna operación relacionada con la autoconfiguración.

Se trata por lo tanto de un estado de espera.

Node down time window

Este nodo proporciona un margen de tiempo cuando el nodo debe recoger las direcciones IP de un nodo que ha abandonado la red.

Más exactamente, el nodo cambia a este estado al detectar que ha perdido la ruta hacia un nodo de cuyo bloque de direcciones es responsable. Esta transición se hace desde cualquier otro estado, ya sea de tipo *ready* o *not_ready*.

Tras el tiempo determinado por el temporizador NODE_DOWN_ASSIGN_TIMER, el nodo volverá al estado de reposo IDLE.

No se debe confundir este temporizador con el NODE_DOWN_TIMER. Aunque se inicien al mismo tiempo al detectar el mismo evento, los procesos involucrados son independientes.

II. Nodo cliente

El procedimiento que sigue un nodo que desea acceder a una red se describe en la siguiente figura:

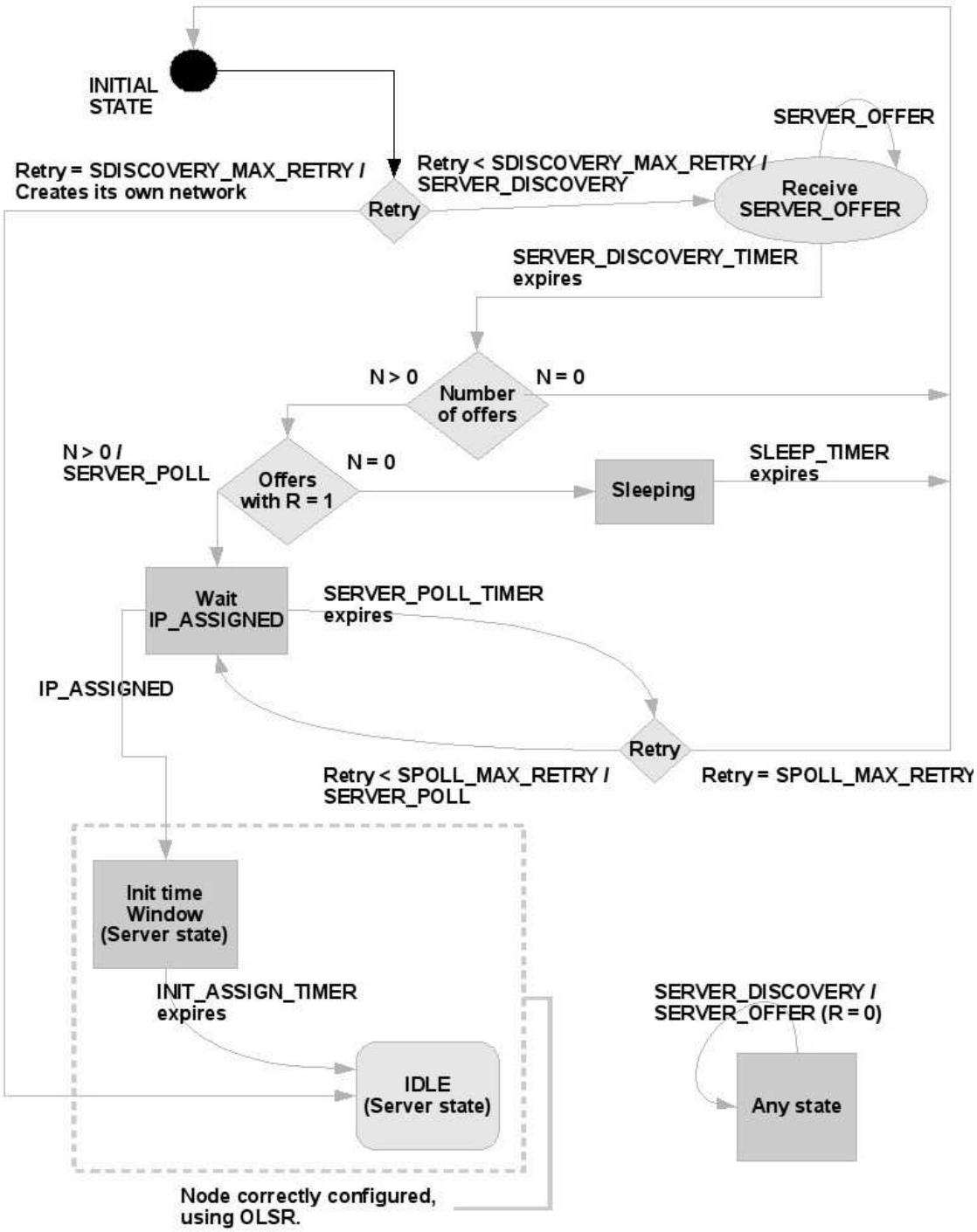


Figura 8.3 – Diagrama de estadod nodo cliente.

El diagrama cuenta con estados de tipo *not_ready*, explicados en el diagrama de estados del nodo servidor y representados con el mismo estilo de rectángulos. Al igual que si se tratase de un nodo ya configurado, el nodo que está en proceso de configuración de su dirección IP responde a peticiones SERVER_DISCOVERY con mensajes SERVER_OFFER. En estos mensajes se da el valor 0 al campo R, ya que el nodo no está en disposición de asignar direcciones IP, solo pretende anunciar su presencia.

INITIAL STATE

Estado inicial, en el que comienza el proceso de autoconfiguración. Si el número de intentos es menor al máximo, SDISCOVERY_MAX_RETRY, entonces el nodo emite un mensaje SERVER_DISCOVERY por cada una de sus interfaces de red que vayan a utilizar la red MANET. Cambia su esado a Receive SERVER_OFFER.

Si se ha llegado al límite de intentos, entonces el nodo desiste de su intención de encontrar una red a la que unirse y crea una nueva. Pasará a ser un nodo servidor, comenzando en el estado IDLE del diagrama de estados del servidor.

Receive SERVER_OFFER

Se trata de un estado de espera, durante el cual se recogen los mensajes SERVER_OFFER de posibles nodos próximos. Al terminar la espera, determinada por el temporizador SERVER_DISCOVERY_TIMER, se procesan las respuestas.

Si no se ha recibido ninguna respuesta SERVER_OFFER, se vuelve al estado inicial. Si hay alguna oferta, se comprueba el número de ellas con el valor 1 en el campo R.

Si de entre las ofertas ninguna tenía el bit R a 1, eso quiere decir que hay nodos cercanos pertenecientes a una red, pero de momento no son capaces de asignar direcciones IP. Por lo tanto, el nodo pasa al estado Sleeping.

Si había alguna oferta con el bit R a 1, se ordenan los servidores por preferencia y se envía un mensaje SERVER_POLL al primero de ellos. En este caso el nodo cambia su estado a Wait IP_ASSIGNED.

Este estado no es de ninguno de los tipos explicados en el diagrama de estados de un nodo cliente. Esto significa que no responde a mensajes SERVER_DISCOVERY, ya que de momento no conoce si hay alguna red cercana a la que unirse.

Sleeping

El nodo pausa sus intentos de entrar en la red durante el tiempo determinado por el temporizador SLEPP_TIMER. Esto es así porque se han recibido mensajes SERVER_OFFER de nodos cercanos que de momento no son capaces de asignar direcciones IP, y lo que se pretende que tras este tiempo ya sean capaces de facilitar la entrada a la red.

Al expirar el temporizador, el nodo volverá al estado inicial.

Wait IP_ASSIGNED

En este estado, el cliente se encuentra a la espera de un mensaje IP_ASSIGNED por parte del servidor al que se envió el mensaje SERVER_POLL.

Si no llega el mensaje esperado, el temporizador SERVER_POLL_TIMER expira. En ese caso, se volverá a intentar enviar un SERVER_POLL al siguiente servidor de la lista generada tras la finalización del temporizador SERVER_DISCOVERY_TIMER. Si se acaba la lista de servidores, o se llega al límite de intentos SPOLL_MAX_RETRY, el nodo vuelve al estado inicial.

Al recibir el mensaje IP_ASSIGNED, el nodo configura su dirección (o direcciones, en caso de disponer de varias interfaces de red). En ese momento, ya participa normalmente en la red, y pasa a ser un nodo servidor.

El estado con el que comienza su comportamiento como servidor es el Init time window, explicado en la sección anterior.

8.3. Resultados y conclusiones

Frente a los protocolos de autoconfiguración basados en la detección de direcciones duplicadas (DAD), el protocolo SARA presenta una gran reducción en la sobrecarga de paquetes de control en la red. En la mayoría de los casos, la configuración se realizará de forma local, es decir, un vecino asignará la dirección al nuevo nodo. Esto implica la emisión de cuatro paquetes de control que no se propagan al resto de la red. En el caso de que no se puedan asignar direcciones localmente se realiza una transmisión *unicast* con el servidor elegido, lo que provoca una sobrecarga mucho menor que un envío *broadcast*. La probabilidad de que no se pueda asignar direcciones localmente depende de la relación entre el número de nodos presentes en la red y el número de direcciones disponibles. En la siguiente gráfica se puede observar el número medio de paquetes de control involucrados en cada proceso de configuración de dirección. En las simulaciones se han utilizado direcciones IP clase C, por lo que disponemos de 254 direcciones de red. Se puede comprobar en la gráfica que cuando existen pocos nodos en la red, el número de mensajes de control necesarios para realizar la configuración se acerca al mínimo, puesto que en la mayoría de los casos se puede realizar la configuración de forma local. Sin embargo, cuando el número de direcciones libres se acerca a 0, no se puede realizar la configuración localmente y se debe recurrir a nodos lejanos para realizar dicha configuración, aumentando el número de mensajes enviados.

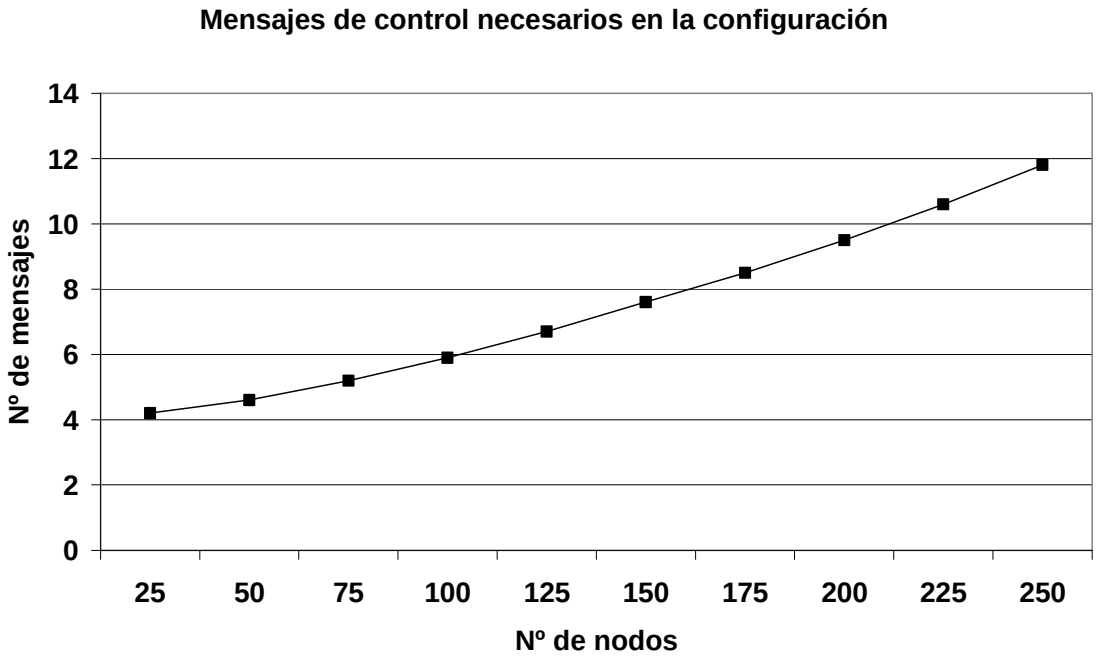


Figura 8.4 – Gráfica de la sobrecarga del protocolo.

A la hora de realizar estas simulaciones se ha mantenido constante el número de entradas en la red por unidad de tiempo. Sin embargo, este factor es importante, sobretodo en redes de alta densidad. Si se producen muchas peticiones de dirección en instantes próximos, el protocolo puede no tener tiempo suficiente para actualizar la tabla de direcciones libres, pudiendo pedir direcciones a un nodo que ya ha asignado sus direcciones libres, degradando así su rendimiento.

En la siguiente gráfica podemos ver la evolución del número de mensajes de control necesarios para asignar una dirección en función del número de peticiones realizadas por segundo. En el caso de la línea superior se ha realizado la simulación en un escenario con 250 nodos. En el otro caso se ha utilizado un escenario de similares características pero con 225 nodos. Se ha empleado una frecuencia de salida de nodos de la red similar a la de entrada para mantener la disponibilidad de direcciones.

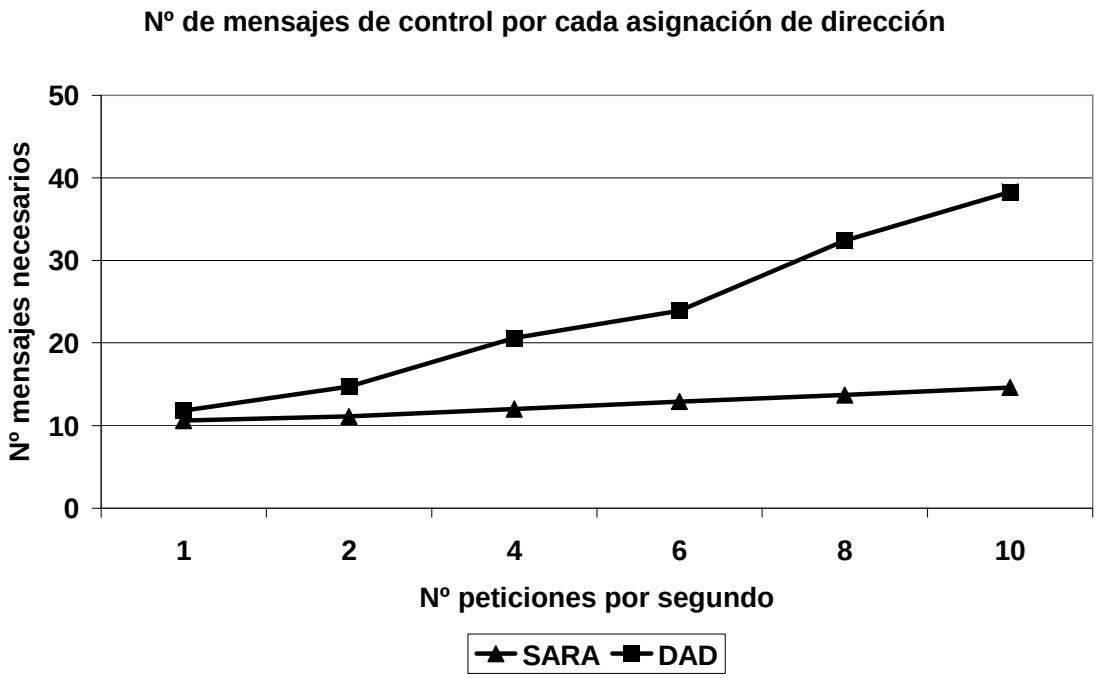


Figura 8.5 – Gráfica comparativa protocolo SARA - DAD.

El principal problema de rendimiento encontrado se da en la situación que refleja esta última gráfica. En situaciones en las que se producen errores en la elección del servidor remoto, la latencia aumenta de forma directamente proporcional al número de mensajes de control enviados. Este incremento en la sobrecarga es aceptable, puesto que sigue siendo menor que la sobrecarga producida por los algoritmos DAD. Sin embargo, en el caso de la detección de direcciones duplicadas se recorre desde el principio toda la red, por lo que la latencia llega a ser menor si la generación aleatoria de direcciones tiene éxito.

Otro de los inconvenientes detectados es el tiempo de reacción ante la salida de los nodos de la red que, en situaciones en las que el número de direcciones libres es pequeño, puede hacer que no se atiendan nuevas peticiones. Por otro lado, hay que mencionar que la distribución de las direcciones IP entre los nodos dependerá exclusivamente de la concentración de estos, y no se realizará de forma igualitaria. De esta forma, algunos nodos pueden tener prioridad a la hora de recibir direcciones. Estos problemas aquí mencionados consituyen la línea de trabajo que a seguir en investigaciones futuras que pretendan mejorar el protocolo SARA.

9. Módulo de seguridad en el protocolo SARA

Este módulo de seguridad se ha desarrollado bajo la dirección del miembro del grupo de investigación GASS Alberto Benito Peral, que ha aportado la elección y planificación de las tareas a realizar así como la formación y asistencia en cuestiones técnicas necesarias. Como apoyo a sus actuales líneas de investigación, lo que se presenta en este capítulo es un módulo que sirve como conexión entre el protocolo SARA y futuras investigaciones que están ya planificadas.

9.1. Introducción

Las redes MANET presentan una larga lista de vulnerabilidades. Su carácter auto-organizado, el uso de un medio compartido y la carencia de infraestructuras provocan que sean susceptibles de recibir diversos ataques como denegación de servicio (DOS), suplantación, ataques contra los mecanismos de encaminamiento, modificación de los mensajes transmitidos, etc.

Prevenir todos los ataques que puede recibir una red MANET es una tarea compleja. Es necesario combinar diferentes técnicas, como monitorización, encriptación o firma digital. En este capítulo se propone un esquema de seguridad que, mediante la cooperación de diferentes mecanismos, permita la detección de ataques, de forma que estos sean contrarrestados y sus iniciadores aislados.

Para poder construir redes seguras, deberemos proporcionar algunos servicios de seguridad que garanticen a los usuarios una transmisión de datos segura.

En primer lugar, debemos proporcionar a los usuarios un mecanismo de autenticación. Es fundamental que los usuarios puedan verificar la identidad de las entidades con quienes comparten información. Puesto que en una red

basada en IP se identifica a cada nodo por su dirección IP, y ésta viene proporcionada por un mecanismo de configuración de direcciones IP, este mecanismo debería realizar esta asignación de forma segura. A la hora de asignar una nueva dirección, se deberá evitar la asignación de direcciones duplicadas y proporcionar al nodo receptor una forma de probar su identidad.

Sin embargo, el proceso de configuración de direcciones puede sufrir varios ataques. Un nodo atacante perteneciente a la red puede asignar direcciones que no están libres, o puede realizar solicitudes de nuevas direcciones para nodos inexistentes, reduciendo el número de direcciones libres. Además, si se asume que los nodos que quieren entrar en la red son bien intencionados, se puede permitir la participación en la red de posibles atacantes.

Aunque a nivel de aplicación los usuarios utilicen mecanismos de seguridad, como encriptación y firmado, para proteger sus datos, pueden sufrir ataques si existen nodos atacantes dentro de la red. Por lo tanto, sería conveniente que se pudiera filtrar la entrada de nodos en la red. Teniendo en cuenta las circunstancias comentadas tanto en esta sección como en el capítulo de seguridad, se ha abordado el diseño de un esquema de seguridad para redes MANET, representado en la figura 9.1.

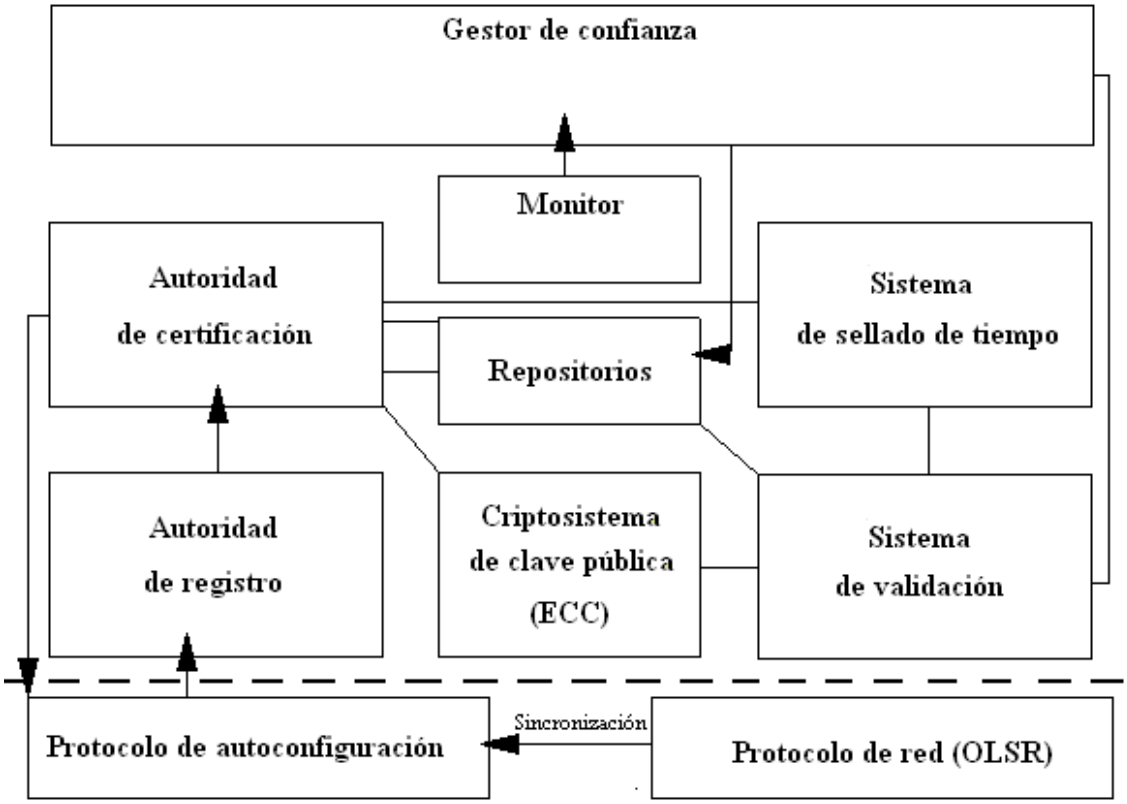


Figura 9.1 – Esquema de seguridad para redes MANET

Dentro de este esquema de seguridad, la labor llevada a cabo en este proyecto ha consistido en la implementación de un criptosistema de clave pública que permite cifrar y firmar mensajes. Este criptosistema puede ser utilizado tanto en la concesión de certificados provisionales en la fase de autoconfiguración, como para cifrar o firmar mensajes de datos y de control durante el tiempo de pertenencia a la red. Puesto que trabajamos con redes formadas por dispositivos con limitaciones de capacidad, es importante que los algoritmos criptográficos sean eficientes. Debido a que los algoritmos criptográficos de clave pública tradicionales son relativamente lentos, se ha trabajado con criptografía basada en curvas elípticas, con el fin de mejorar estas prestaciones. En las siguientes secciones describiremos el trabajo realizado para conseguir dicha implementación, así como la propuesta realizada para el formato de certificados.

9.1.1 Contexto

Como se ha indicado anteriormente, el módulo de seguridad implementado sirve como base a otros mecanismos más complejos. Para entender su utilidad, en esta sección se expone el contexto en el que se enmarca. Lo que sigue es un esbozo del proyecto futuro planificado en el grupo de investigación GASS, que usará como base el trabajo presentado en esta memoria.

Puesto que las redes MANET pueden aplicarse en diversos escenarios, teniendo cada uno unos requisitos de seguridad propios, se hace necesario crear el concepto de nivel de seguridad de la red.

Se pueden definir nuevas redes con niveles de seguridad que se encuentren en el intervalo [0, 1]. Una red con un nivel de seguridad igual a 0 se denominará red pública, es decir, permitirá el acceso a cualquier nodo si existen direcciones libres. En el caso de que el nivel de seguridad sea 1, hablaremos de una red privada, a la que sólo podrán acceder determinados nodos. Se podrá seleccionar como condición de acceso el conocimiento de una contraseña o la posesión de un certificado cuyas características las especificará el precursor de la red.

Adicionalmente, se podrán definir diferentes niveles de seguridad que restrinjan el acceso a la red en función de distintos parámetros. Estos niveles recibirán valores entre 0 y 1, incluyendo en los parámetros de acceso los parámetros que deben comprobarse y su valor umbral. Por lo tanto, cuando un nodo crea una nueva red, esta red vendrá definida por los siguientes atributos:

- Identificador de red.
- Nivel de seguridad.
- Modo de acceso.
- Parámetros de acceso.

Cada nodo almacenará en su módulo de seguridad la información relativa a la red a la que pertenece, para que esta pueda ser consultada por cada mecanismo de seguridad.

Cuando un nuevo nodo solicite la entrada en la red solicitando una dirección IP, el nodo que actúa como servidor se encargará de preguntar al módulo que ejerce de autoridad de registro si el nuevo nodo puede acceder a la red. En caso afirmativo se le concederá un certificado provisional firmado por el propio nodo servidor.

El motivo de que este certificado sea provisional es que un nodo que ha conseguido entrar en la red de forma ilegítima podría autorizar la entrada de otros nodos. La manera de evitar este riesgo es hacer que para que un nodo pueda participar de forma continua en la red deberá estar en posesión de un certificado firmado de manera conjunta.

El proceso de firma conjunta se realizará de la siguiente manera: Los gestores de confianza de los distintos nodos que forman la red cooperarán para crear grupos de confianza, de forma que cada nodo de la red pertenezca a uno de dichos grupos. Para que un certificado sea válido será necesario que un representante de cada grupo firme el certificado. Para conseguir esto, el módulo de certificación del nodo que ha ejercido como servidor, después de conceder el certificado provisional, consulta la tabla en la que aparecen los grupos de confianza que forman la red. Esta tabla se encontrará dentro del repositorio de datos. A continuación, elige representantes de cada grupo y les envía una solicitud de firma. Cuando recibe la contestación de todos los grupos envía el certificado definitivo al nodo cliente.

Cuando un nodo detecte a otro nodo de cuya existencia no tenía constancia, solicitará su certificado antes de comenzar a trabajar de forma conjunta. Una vez recibido el certificado, comprobará su validez y en el caso de que sea válido lo almacenará en una tabla dentro del repositorio.

Para vigilar el comportamiento de los nodos vecinos se puede activar de forma adicional el mecanismo de monitorización. Este mecanismo trabajará como sniffer para detectar comportamientos anómalos en la red, como la no

retransmisión de paquetes de datos o de control, la redirección incorrecta de mensajes, etc. Cuando el monitor detecte comportamientos anómalos lo notificará al gestor de confianza. Los respectivos gestores de confianza de los nodos de la red compartirán información sobre los comportamientos anómalos que se producen, y en el caso de que se observen actuaciones malintencionadas de forma continuada o de carácter grave, los gestores de confianza pueden, de manera conjunta, iniciar un procedimiento de expulsión de la red.

9.2. Criptografía de curva elíptica

La criptografía de clave pública se basa en la existencia de problemas matemáticos cuyo cómputo es sencillo, pero para los cuales no existe ningún algoritmo eficiente que realice la operación inversa. Por ejemplo, RSA, cuyo uso está muy extendido en nuestros días, basa su seguridad en la intratabilidad del problema del cálculo del logaritmo discreto sobre cuerpos finitos.

Estos sistemas permiten que se pueda transmitir información de forma segura sin necesidad de disponer de un canal seguro para intercambiar las claves. En el caso de las claves simétricas, si no se dispone de un canal seguro, al transmitir el valor de nuestra clave a un vecino, cualquier atacante podría interceptar el envío de la clave y, a partir de entonces, descifrar los mensajes. En el caso de la criptografía asimétrica, no existe ningún riesgo al dar a conocer una clave pública. Si un nodo que quiere comunicarse con otro nodo cifra el mensaje con la clave pública, la única forma de descifrarlo será utilizando la clave privada, que no se comparte con nadie. Puesto que calcular la clave privada a partir de la clave pública es un problema intratable, no se podrá descifrar el mensaje original.

Sin embargo, estos algoritmos presentan la desventaja de ser muy costosos computacionalmente. Necesitan trabajar con números muy grandes para ser seguros, en torno a 1024 bytes. El procedimiento de generación de claves es costoso, y tanto el cifrado como el descifrado de mensajes es mucho más lento que en el caso de los algoritmos de cifrado simétrico. Por este motivo, se están

llevando a cabo investigaciones para diseñar nuevos criptosistemas de clave pública más eficientes.

Dentro de estos nuevos criptosistemas hay que destacar los criptosistemas basados en curvas elípticas, cuya importancia ha ido creciendo durante los últimos años, llegando en la actualidad a formar parte de los estándares industriales. Si bien se han diseñado variantes con curvas elípticas de criptosistemas clásicos, como el RSA, su principal logro se ha conseguido en los criptosistemas basados en el problema del logaritmo discreto, como los de tipo ElGamal. En este caso, los criptosistemas elípticos garantizan la misma seguridad que los construidos sobre el grupo multiplicativo de un cuerpo finito primo, pero con longitudes de clave mucho menores.

Hoy en día el uso generalizado de redes informáticas en el tratamiento y transmisión de la información, así como el aumento constante del número de usuarios de estos sistemas, han motivado la necesidad de mejorar la seguridad en las comunicaciones. Son muchas y variadas las situaciones donde cabe garantizar la privacidad, la integridad o la autenticación de la información transmitida. Tales necesidades se han podido satisfacer mediante el uso de distintos protocolos criptográficos, en los que se combinan a menudo criptosistemas de clave compartida con criptosistemas de clave pública. Como ya se expuso anteriormente, uno de los inconvenientes de la criptografía de clave compartida es la distribución de claves, o mejor dicho, la necesidad que dos usuarios tienen de pactar previamente su clave secreta si desean cifrar los mensajes que se van a enviar. Para solventar este problema, se propone el intercambio seguro de claves entre dos usuarios. Éste es el primer paso que origina la criptografía de clave pública, en tanto que no va a ser necesaria la comunicación directa entre los usuarios. Bastará con emplear cierta información que el receptor va a hacer pública y, por tanto, accesible para cualquier entidad que se quiera comunicar con él. La seguridad de un criptosistema de clave pública reside entonces en problemas matemáticos subyacentes que se conjeturan computacionalmente difíciles, es decir, problemas para los que no se conocen algoritmos eficientes para resolverlos.

Entre los criptosistemas de clave pública más usados actualmente podemos citar el sistema RSA, debido a Rivest, Shamir y Adleman en 1977, basado en el problema de la factorización de enteros, y el criptosistema ElGamal basado en el problema del logaritmo discreto sobre el grupo multiplicativo de un cuerpo finito.

Sin embargo, tanto el avance en la eficiencia de los nuevos algoritmos de factorización (algoritmo de *Lenstra* [41] basado en curvas elípticas y la criba en cuerpos de números), como en el criptoanálisis del problema del logaritmo discreto (método del *Index-Calculus*) han provocado la necesidad de aumentar el tamaño de las claves usadas. Los criptosistemas basados en curvas elípticas aparecen como una alternativa a los criptosistemas antes mencionados, tanto por la disminución del tamaño de las claves que se requieren, manteniendo la misma seguridad computacional, como por el amplio abanico de grupos que ofrecen sobre el mismo cuerpo base. Su implantación en algunos sistemas de telecomunicaciones o tarjetas inteligentes es un hecho contrastado, que parece aumentar día a día debido a las ventajas de estos criptosistemas.

La criptografía de curva elíptica se basa en el problema del logaritmo discreto (descrito en el apartado de seguridad en redes MANET). Por ejemplo, encontrar el valor de b dada la ecuación ab = c, cuando a y c son valores conocidos, puede ser un problema de complejidad exponencial para ciertos grupos finitos de gran tamaño; mientras el problema inverso, la exponenciación discreta puede ser evaluado eficientemente usando por ejemplo exponenciación binaria.

9.2.1. Fundamentos de las curvas elípticas

Antes de describir cómo se puede diseñar un criptosistema asimétrico basándose en las propiedades de las curvas elípticas, se introducirá el concepto matemático de curva elíptica y alguna de sus propiedades.

Una curva elíptica sobre un cuerpo **K** es una curva algebraica sin puntos singulares que viene dada por una ecuación del tipo

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4 + a_6,\ a_i \in \mathbf{K},$$

denominada ecuación general de *Weierstrass*.

En el caso de que el cuerpo K sobre el que se define la curva no sea ni 2 ni 3 (la característica de un cuerpo se define como el número entero positivo más pequeño n tal que $n \cdot 1 = 0$, o 0 si no existe tal n), la ecuación se puede expresar como

$$y^2 = x^3 + ax + b, \qquad a,b \in \mathbf{K},$$

denominada ecuación reducida de Weierstrass, debiendo ser el discriminante del polinomio cúbico en x no nulo, es decir, $4a^3 + 27b^2 \neq 0$, para que la curva no tenga singularidades.

Si E/**K** es una curva elíptica sobre un cuerpo **K**, denotaremos por E(**K**) el conjunto de puntos $P = (x, y) \in \mathbf{K} \times \mathbf{K}$ que satisfacen la ecuación de la curva junto con el punto del infinito de la curva O.

Sobre el conjunto de puntos de una curva elíptica E/K se puede definir una operación interna que dota a E(**K**) de una estructura de cuerpo abeliano. Esta operación binaria se denomina suma elíptica. Esta operación interna se define mediante el llamado método de la cuerda y la tangente (*Figura 9.2*). Este método consiste en considerar la recta *r* que pasa por los dos puntos *P* y *Q* (en el caso que coincidan se toma la tangente la curva en el punto) y calcular el tercer punto de intersección *R* de la recta *r* con la curva. El punto *P* + *Q* es el punto intersección de la curva con la recta que pasa por *R* y el punto del infinito, es decir, la recta que pasa por *R* y es paralela al eje de ordenadas. El elemento neutro del conjunto E(**K**) respecto a la suma elíptica es el punto del infinito.

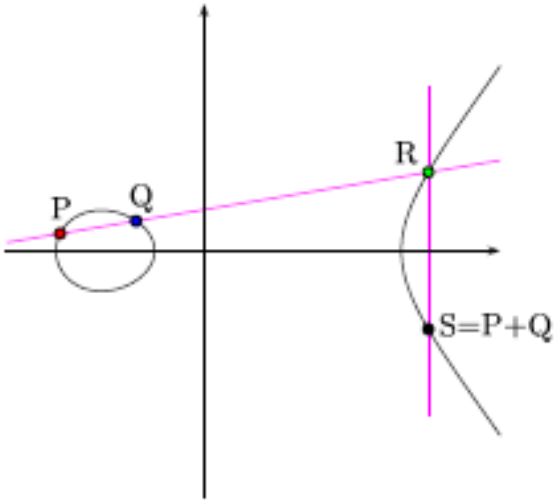


Figura 9.2 – Ejemplo de suma elíptica

De forma analítica, se pueden expresar las coordenadas del punto $P + Q = (x3, y3)$ en función de P (x1, y1) y Q (x2, y2) si $P + Q \neq O$:

$$x_3 = \lambda^2 - x_1 - x_2,$$

$$y_3 = (x_1 - x_3)\,\lambda - y_1,$$

donde $\lambda = (y_1 - y_2)/(x_1 - x_2)$ si $x_1 \neq x_2$ y donde $\lambda = (3x_1^2 + a) / 2y_1$ en caso de $x_1 = x_2$ e $y_1 \neq 0$. El simétrico de un punto $P = (x, y)$, que denotamos por $-P$, es el punto de coordenadas $(x, -y)$.

Una vez definida esta operación, se puede definir dado un punto P y un natural k, el punto $k \cdot P$ como $P + \ldots + P$. Esta operación es el equivalente aditivo a la exponenciación en grupos abelianos multiplicativos. La operación inversa se puede definir como $\log_P (k \cdot P) = k$. Estas operaciones son la base de la criptografía con curvas elípticas.

Se pueden definir curvas elípticas sobre cualquier cuerpo. Las curvas elípticas definidas sobre un cuerpo finito $\mathbf{F}_q$, con $q = p^m$ y p primo, presentan algunas propiedades especiales.

En primer lugar, el Teorema de Hasse nos da información respecto al cardinal del conjunto de puntos de la curva, $\#E(\mathbf{F}_q)$. H. Hasse enunció que el

cardinal de una curva elíptica definida sobre $\mathbf{F}_q$ será cercano q, en concreto, el valor de $\#E(\mathbf{F}_q)$ estará dentro del intervalo $[q + 1 – 2\sqrt{q}, q + 1 + 2\sqrt{q}]$

Por otro lado, tenemos que, el orden de un punto $P \in E(\mathbf{F}_q)$ será un natural k, tal que $k \cdot P = O$.

Respecto a la estructura del conjunto de puntos $E(\mathbf{F}_q)$, J. W. Cassels obtuvo el siguiente resultado, que caracteriza dicho conjunto: “El grupo es isomorfo al grupo cíclico $\mathbf{Z}_m$, donde $m = \#E(\mathbf{F}_q)$, o bien al grupo $\mathbf{Z}_{m1} \times \mathbf{Z}_{m2}$, donde $m_1 \cdot m_2 = m$, $m_2 | m_1$ y $m_2 | (q - 1)$”.

Recíprocamente, E. Waterhouse demostró que sobre un cuerpo finito primo $\mathbf{F}_p$ existen curvas elípticas con cardinal cada uno de los valores del intervalo de Hasse y estructura de grupo cada una de las posibles estructuras dadas por Cassels.

9.2.2. Seguridad con curvas elípticas

Como se ha mencionado anteriormente, la seguridad en un criptosistema de clave pública se basa en la imposibilidad de resolver un problema de inversión. En el apartado anterior se ha definido, en primer lugar, la operación cerrada denomina suma elíptica. Posteriormente, se ha definido la operación $k \cdot P$ con k un número natural y P un punto de la curva. Esta operación se puede Implementar de forma eficiente empleando el algoritmo de exponenciación binaria. Sin embargo, conociendo los puntos Q y P, siendo $Q = k \cdot P$, el cálculo de k no es tan sencillo. Este problema se conoce como el Problema del Logaritmo Discreto sobre Curvas Elípticas (PLDCE).

La mayoría de los algoritmos que resuelven el Problema del Logaritmo Discreto tienen un coste exponencial. Entre ellos destacamos el algoritmo *Pohlig-Hellman* [42], paso de niño – paso de gigante, ρ de Pollard, etc. Además, en el caso del grupo multiplicativo $\mathbf{F}_p^*$ existe un algoritmo que puede resolver este problema en tiempo subexponencial, llamado *Index-Calculus*. Sin embargo, este algoritmo no es aplicable cuando trabajamos con curvas elípticas definidas sobre $\mathbf{F}_p$. Por tanto, tenemos que los algoritmos existentes

para realizar el cálculo del logaritmo discreto sobre curvas elípticas tienen coste exponencial, lo que nos permite utilizar esta operación como base de un criptosistema de clave pública. Sin embargo, estos algoritmos pueden obtener resultados de forma rápida en determinadas curvas elípticas. Por tanto, para poder garantizar la seguridad de nuestro sistema, se deberán descartar las curvas que permitan el cálculo de esta operación de forma eficiente.

Para que una curva elíptica sobre un grupo $\mathbf{F}_p$ sea criptográficamente útil, se debe cumplir que el cardinal de los puntos $E(\mathbf{F}_p)$ sea de la forma $f \cdot q$, con q primo y f un entero pequeño. En el caso contrario, sería vulnerable al ataque de Pohlig-Hellman.

De la misma manera, habrá que descartar las curvas que sean supersingulares y anómalas. Las curvas supersingulares son aquéllas que tienen cardinal $p + 1$, en cuyo caso son vulnerables al ataque MOV (Ataque sobre el logaritmo discreto). Las curvas anómalas son las que tienen cardinal p y, aunque son resistentes al ataque MOV, se puede dar un algoritmo polinómico para resolver el PLD sobre su grupo de puntos.

Además de lo comentado anteriormente, la propiedad de las curvas elípticas definidas sobre cuerpos finitos $\mathbf{F}_p$ enunciada por E. Waterhouse, dificulta el proceso de cálculo del PLD. Puesto que existen curvas elípticas con cardinal cada uno de los posibles enteros en el intervalo de Hasse, tendremos que existen hasta $4\sqrt{p}$ cardinales diferentes para cada cuerpo $\mathbf{F}_p$.

Por tanto, si se generan curvas elípticas sobre cuerpos finitos (con p un primo suficientemente grande como para que el problema del PLD sea intratable) y que cumplan los requisitos mencionados anteriormente para que sean criptográficamente útiles, se podrá crear un criptosistema de clave pública. Este criptosistema se basará en la multiplicación de puntos de la curva elíptica por un número natural. De esta forma, se elige un punto base G específico y publicado para utilizar con la curva E(q). Se escoge un número entero aleatorio k como clave privada, y entonces el valor P = k*G se da a conocer como clave pública (nótese que la supuesta dificultad del problema del logaritmo discreto implica que k es difícil de deducir a partir de P). De esta

manera si dos nodos, A y B tienen las claves privadas kA y kB, y las claves públicas PA y PB, entonces el nodo A podría calcular kA*PB = (kA*kB)*G; y el nodo B puede obtener el mismo valor dado que kB*PA = (kB*kA)*G.

Esto permite establecer un valor "secreto" que tanto un nodo como el otro pueden calcular fácilmente, pero que es muy complicado de derivar para una entidad ajena. Además, el nodo B no consigue averiguar nada nuevo sobre kA durante ésta transacción, de forma que la clave del nodo A sigue siendo privada.

Los métodos utilizados en la práctica para cifrar mensajes basándose en este valor secreto consisten en adaptaciones de antiguos criptosistemas de logaritmos discretos originalmente diseñados para ser usados en otros grupos. Entre ellos se podrían incluir Diffie-Hellman, ElGamal y DSA.

9.2.3. El Gamal Elíptico

El teorema de ElGamal propone un criptosistema de clave pública basado en la intratabilidad del problema del logaritmo discreto (PLD):

Dado un grupo finito cíclico G, un generador *g* de G y un elemento *x* de G, encontrar el entero n tal que $x = g^n$, es decir, el logaritmo discreto de *x* en base *g*.

Nótese que conocidos *g* y *n*, se puede calcular $x = g^n$ de manera eficiente, pero en cambio, conocidos *g* y *x*, encontrar *n* es un problema computacionalmente difícil.

Para garantizar la seguridad del criptosistema ElGamal sobre el grupo multiplicativo $\mathbf{F}_p$ * hay que considerar primos *p* cuyo tamaño sea suficientemente grande para que los algoritmos conocidos que resuelven el PLD no sean eficientes. Así, el primo *p* se tiene que elegir de manera que *p*-1 tenga un factor primo grande. En caso contrario, se podría aplicar el método de Pohlig- Hellman, que reduce el PLD en $\mathbf{F}_p$ * a grupos cuyos órdenes son los factores de *p* - 1. Tanto este método como todos los que se conocen de índole general (paso de niño - paso de gigante, ρ de Pollard,...) en el sentido que se

pueden aplicar a cualquier grupo, tienen costo exponencial. Sin embargo, en el caso particular grupo Fp * existe un método, basado en expresar un elemento del grupo como producto de elementos de una cierta base, denominado Index-Calculus, que tiene coste subexponencial.

La importancia, pues, de considerar el criptosistema ElGamal sobre el grupo de puntos de una curva elíptica E definida en un cuerpo Fq, q = p o q = 2m, reside en que los algoritmos que resuelven el PLD sobre E (F_q) tienen coste exponencial y, por tanto, el tamaño de las claves puede ser mucho menor (160 bits versus 1024).

Para la configuración de un criptosistema necesitamos generar un primo p para definir el cuerpo $\mathbf{F}_p$, los parámetros *a* y *b* de la curva *E* sobre $\mathbf{F}_p$ y un punto *P* de la curva cuyo orden sea un entero *n* que tenga un factor primo del tamaño de *p*. Trabajaremos entonces en el subgrupo de Ea,b ($\mathbf{F}_p$) de orden n generado por *P*. Como clave privada se elige un entero d en el intervalo [1, *n* - 1] y la clave que se hace pública es el punto $Q = d \cdot P$ de la curva. El mensaje que se quiere cifrar, suponiendo que se ha convertido en un número natural *m*, 0 < *m* < *p*, se identifica habitualmente con la abscisa de un punto *M* de la curva (en caso que $m^3 + a \cdot m + b$ no sea un cuadrado en $\mathbf{F}_p$ se toma $m+1$.

El algoritmo del criptosistema ElGamal elíptico basa su seguridad en el problema del Logaritmo Discreto Elíptico, que se expone a continuación:

Dada una curva elíptica *E*, y un grupo $\mathbf{F}_q$, se considera el grupo abeliano de puntos racionales *E* ($\mathbf{F}_q$) de la forma (*x*, *y*), donde *x*, *y* Є $\mathbf{F}_q$, y donde el grupo operación "+" se define en esta curva elíptica. Se define entonces una segunda operación "*" | Z×E (q) → E (q): si P es algún punto en E (q), entonces se define 2**P* = *P* + *P*, 3**P* = 2**P* + *P* = *P* + *P* + *P*, y así en adelante. Nótese que dados los enteros *j* y *k*, *j**(*k***P*) = (*j***k*)**P* = *k**(*j***P*). El problema del logaritmo discreto de una curva elíptica (PLDCE), es pues, determinar el entero *k*, dados los puntos *P* y *Q*, cuando *k***P* = *Q*. Se cree que el típico problema del logaritmo discreto sobre el grupo multiplicativo (PLD) y el PLDCE no son problemas equivalentes, y que el PLDCE es significativamente más difícil que el PLD. En el uso criptográfico, se elige un punto base *G* específico y publicado para

utilizar con la curva $E(\mathbf{F}_q)$. Se escoge un número entero aleatorio k como clave privada, y entonces el valor $P = k*G$ se da a conocer como clave pública (nótese que la supuesta dificultad del PLDCE implica que k es difícil de deducir a partir de P).

Algoritmo (Cifrado criptosistema ElGamal elíptico)

Entrada: Los parámetros (p, a, b, G, n), la clave pública Q y el mensaje en claro m.

Salida: El mensaje cifrado (C1, C2).

- Representar el mensaje m como un punto M de Ea,b (Fp).
- Escoger un entero aleatorio r en [1, n-1].
- Calcular los puntos C1 =r ·P y C2 = M + r · Q en Ea,b (Fp).
- Devolver (C1, C2).

Algoritmo (Descifrado criptosistema ElGamal elíptico)

Entrada: Los parámetros (p, a, b, G, n), la clave privada d y el mensaje cifrado (C1, C2).

Salida: El mensaje en claro m.

- Calcular los puntos d · C1 = d · r · P= r · Q y M =C2 - r · Q en Ea,b (Fp).
- Obtener el mensaje en claro m del punto M.
- Devolver m.

Se ha expuesto como puede utilizarse para cifrar y descifrar mensajes El Gamal Elíptico. Sin embargo, para poder construir nuestro PKI se necesita también un algoritmo de firma que permita garantizar la identidad del remitente de los mensajes recibidos. El algoritmo DSA (*Digital Signature Algorithm*) es una variante de la firma de El Gamal, que es la base del estándar DSS (*Digital*

Signature Standard). El algoritmo ECDSA (*Elliptic Curve Digital Signature Algorithm*) es el análogo al algoritmo DSA con curvas elípticas. Los procesos de generación y verificación de firma, considerando los mismos parámetros que en la configuración del criptosistema de El Gamal, son los siguientes:

Algoritmo (Generación de firma digital ECDSA)

Entrada: Los parámetros (*p, a, b, P, n*), la clave pública *Q*, la clave privada *d* y el mensaje en claro *m*.

Salida: El mensaje m con la firma (*r*, *s*).

- Calcular el Hash del mensaje h = H (m).
- Escoger un entero aleatorio k en [1, n - 1].
- Calcular el punto k · P = (x, y) en Ea,b (Fp).
- Calcular r = x (mod n) (si r = 0 ir al inicio).
- Calcular s = k-1(h + d · r) (mod n) (si s = 0 ir al inicio).
- Devolver m y (r, s).

Para verificar la firma a partir del Hash del mensaje hay que calcular el inverso *w* de *s* módulo *n*. Entonces con la clave pública basta calcular el punto $R = (w \cdot h) \cdot P + (w \cdot r) \cdot Q$ y comprobar, dado que $k = w \cdot h + w \cdot d \cdot r$ y $Q = d \cdot P$, que las abscisas de los puntos *R* y $k \cdot P$ coinciden.

9.2.4. Implementación y resultados

Para poder evaluar el rendimiento de EL Gamal Elíptico y la viabilidad de su aplicación en redes MANET, se implementó el algoritmo en C++, así como los criptosistemas RSA y El Gamal. Posteriormente, se ha empleado esta implementaicón para el firmado de mensajes en las simulaciones realizadas en el simulador NS-3. La implementación de este criptosistema presenta mayores dificultades que otros criptosistemas clásicos. En esta sección se describe el trabajo realizado y las ventajas obtenidas frente a RSA y El Gamal.

Para implementar estos criptosistemas se ha implementado previamente la clase *BigUInteger* que permitirá trabajar con enteros positivos de cualquier longitud. A la hora de implementar esta clase y sus métodos y operadores se ha sido especialmente cuidadosos en el uso de memoria y en la eficiencia de los algoritmos. Se ha intentado reducir el consumo de memoria al máximo y emplear los algoritmos más eficientes para realizar cada operación. Se han implementado los operadores básicos, así como algunas operaciones de especial interés en las aplicaciones criptográficas. Entre las operaciones que se pueden realizar con la clase *BigUInteger* se encuentran las operaciones aritméticas.

A partir de esta clase se han implementdo tres criptosistemas para poder comparar sus prestaciones: RSA, El Gamal y El Gamal Elíptico. Esta seccción se centrará en la implementación de El Gamal Elíptico, puesto que era el objetivo final del trabajo.

Para poder implementar este criptosistema se necesita, en primer lugar, definir un cuerpo finito Z_p, con p primo de 160 bits. El primer problema de implementación es, al igual que en RSA y El Gamal, la generación de primos. La distribución de los números primos es un problema aún por resolver, por lo que no existe un algoritmo que pueda generar números primos de forma directa. Por lo tanto, la forma de generar números primos en un intervalo es recorrer dicho intervalo y comprobar si cada número es primo o no. Por lo tanto, el problema de generar números primos se reduce a encontrar un algoritmo eficiente que verifique la primalidad de un número. Hasta el año 2002 no se había encontrado un algoritmo de coste no exponencial que garantizase la primalidad de un número entero. Este algoritmo determinista se conoce como análisis de primalidad AKS [43] (de Manindra Agrawal, Neeraj Kayal y Nitin Saxena) y puede decidir si un número es primo o compuesto en un tiempo proporcional a $\log^k n$. Sin embargo, la constante k es demasiado grande y hace difícil la aplicación de este algoritmo en la práctica. Por tanto, se ha decidido trabajar con el test probabilístico *Miller-Rabin* [44] (ver anexo). Este algoritmo verifica la primalidad de un número con cierto grado de certeza. Se estima que en cada iteración del algoritmo, la probabilidad de que un número compuesto sea considerado “probable primo” es ¼. Por tanto, la probabilidad de error tras

k iteraciones es de $1/(4^k)$. En la práctica, la probabilidad de error es aún menor, por lo que este algoritmo se considera suficiente para verificar la primalidad de un número, siendo más rápido que el análisis AKS. Veamos un resumen de los resultados obtenidos con el *Test de Miller-Rabin* para los números enteros positivos menores de 20000 y que muestran la eficacia de dicho algoritmo.

Número de iteraciones	**Errores cometidos**
1	7351
2	546
3	57
4	3
5	0
6	0
7	0
8	0
9	0

Una vez generado el número primo, se deberen asignar valores a los parámetros *a* y *b* de la ecucación de la curva elíptica. En este caso, el problema se reduce a generar dos enteros de forma aleatorio que cumplan la condición $4a^3 + 27b^2 \neq 0$. Una vez definida una curva se debe calcular su cardinal. El cálculo del cardinal permite comprobar la curva es criptográficamente útil o no. En caso negativo se debe probar con *a* y *b* diferentes. Calcular el cardinal de la curva es un problema complejo, y los algoritmos conocidos presentan coste polinómico. Los algoritmos más conocidos son los algoritmos de *Shank* y de *Schoof*. Además de los problemas de eficiencia respecto al tiempo, estos algoritmos presentan problemas de consumo de memoria. Sin embargo, estos problemas se pueden paliar combinando dichos algoritmos. Ambos algoritmo y sus dos combinaciones pueden consultarse en el anexo correspondiente.

Longitud del primo	Tiempo		
	Shank	**C 1**	**C 2**
$\approx 10^{12}$	1 min 24 s	2 min 58 s	2 min 38 s
$\approx 10^{15}$	9 min 10 s	5 min 47 s	2 min 22 s
$\approx 10^{16}$	1 h 2 min	9 min 17 s	2 min 50 s
$\approx 10^{17}$	57 min 2 s	15 min 19 s	4 min 51 s
$\approx 10^{18}$	-	28 min 23 s	15 min 18
$\approx 10^{19}$	-	1 h 3 mon	25 min 12
$\approx 10^{20}$	-	51 min 10 s	21 min 7 s

En esta tabla se muestra que el algoritmo de *Shank* comienza a tener problemas con longitudes demasiado grandes. Estos problemas se deben al alto consumo de memoria. Además de solucionar los problemas de consumo de memoria, la combinación con el algoritmo de *Schoof* reducen el tiempo de ejecución de forma notable cuando la longitud del primo crece. Sin embargo, estos tiempos dificultan su aplicación en la práctica, especialemente en entornos con capacidad computacional limitada. En los últimos años se han hecho algunas propuestas basadas en resultados obtenidos por el matemático P. L. Sylow, que no han sido analizadas hasta la fecha en el marco de este proyecto, pero que serán analizadas en el futuro.

Una vez calculado el cardinal de la curva se genera un punto perteneciente a dicha curva. Para generar dicho punto generamos un valor aleatorio para x, tal que $1 < x < p$. Se comprueba que existe la raíz cuadrada módulo p del valor v dado por $v = x^3 + ax + b$. Para comprobar la existencia de raíces se utilizó el símbolo de Legendre. Para realizar el cáculo de la raíz cuadrada modular se debe implementar el cálculo del símbolo de Jacobi.

Una vez obtenido el punto, sólo se debe generar un número entero positivo aleatorio k, de forma que posteriormente se calcule $Q = k \cdot P$. Con esto ya se tienen el par de claves, y se pueden realizar operaciones de cifrado y de firma digital.

Debido a que el mejor algoritmo, actualmente conocido, para la resolución del problema en el que basa su seguridad ECC es de orden completamente exponencial, a diferencia de los métodos usados para romper criptosistemas como RSA cuyo orden es sub-exponencial, se puede lograr el mismo nivel de seguridad que otros métodos con longitudes de claves menores. Ejemplo de esto es mostrado en la siguiente tabla:

Tiempo para romper el sistema en MIPS-años	**RSA Tamaño de clave**	**ECC Tamaño de clave**
10^4	512	106
10^8	768	182
10^{11}	1024	160
10^{28}	2048	210
10^{38}	21000	600

Esta disminución en la longitud de la clave que se gana al usar ECC produce como resultado que los algoritmos de encriptación y desencriptación sean más rápidos, pero a costa de una menor protección contra ataques de fuerza bruta.

Uno de los puntos que los expertos le imputan como debilidad a ECC es que el tiempo en el que ha salido a la palestra pública es bastante corto en comparación con sistemas ampliamente probados como RSA.

Otro punto en contra de ECC es que es un criptosistema bastante complejo. Sus implementaciones pueden ser un punto problemático, ya que es necesario poseer un amplio conocimiento en teoría de grupos y curvas elípticas para realizar una implementación confiable y eficiente de dicho sistema. Esto en

comparación con RSA es una desventaja, pues la manera como RSA realiza la encripción es mucho más sencilla lo que permite que personas con conocimiento medio de criptografía puedan realizar implementaciones confiables del algoritmo.

Para aplicar la criptografía con ECC a las redes MANET se ha tenido el principal problema en el coste de los algoritmos del cálculo del cardinal de las curvas elípticas. Para compensar este problema, s han tomado algunas medidas. En primer lugar, se ha reducido la longitud de clave. En un principio, se pensó utilizar claves de 160 bits, pero finalmente se han empleado claves de 106 bits. Realizar un ataque contra un sistema con claves de 106 bits sigue siendo costoso. Sin embargo, para conseguir mayores garantías de seguridad, se generan nuevas claves después de un intervalo de tiempo, para evitar que en caso de que se obtenga la clave privada ésta no sea válida. Por otro lado, para acelerar los cálculos, se comparten los parámetros de la curva elíptica. El nodo precursor de la red genera p, a y b, y calcula el cardinal de la curva. Posteriormente comparte esta información con los nuevos participantes. De esta forma, los nuevos participantes no deben realizar el costoso cálculo del cardinal de la curva. El conocimiento de esta información no hace viable el cálculo del logaritmo discreto.

9.3. Autoridad de registro y certificación

En esta apartado se tratará el bloque de la autoridad de registro y la autoridad de certificación dentro del entramado de la seguridad del protocolo SARA.

9.3.1. Descripción

La autoridad de registro corresponde al nodo servidor en un proceso de configuración de un nodo cliente. De esta manera el nodo servidor se encarga de proporcionar una garantía de autenticidad al nodo cliente para que pueda interactuar con el resto de la red con total seguridad.

La autoridad de registro certifica la entrada de un nuevo nodo gracias a la dirección IP del nodo cliente y su clave pública. En el proceso de autoconfiguración del nodo cliente, éste crea una clave pública adecuada en el momento en que elige un nodo servidor que lo configure. En este momento envía un mensaje SERVER_POLL al nodo servidor en el cual encapsula la clave pública generada. De esta manera el nodo servidor, o autoridad registradora asocia el nuevo nodo a esta clave pública. El nodo servidor envía un permiso (certificado) temporal al nodo cliente que se está configurando, el cual será usado por parte del nodo cliente hasta que la autoridad certificadora le asigne un certificado fijo.

La autoridad de certificación consiste en una unidad distribuida constituida por un grupo de nodos pertenecientes a diferentes subgrupos que conjuntamente estiman si el certificado de un nodo es válido, o si por el contrario deben desechar su petición de acceso a la red. Para la realización de la comprobación por parte de la autoridad certificadora el certificado definitivo está formado por codificaciones de claves de diferentes nodos (cada uno perteneciendo a un subgrupo de la red).

Para el buen funcionamiento de este modelo de seguridad se tiene en cuenta que en la red se conocen las claves públicas de todos los nodos pertenecientes a dicha red.

9.3.2. Formato de los certificados

Los certificados dotan a los nodos receptores de una forma de identificarse dentro de la red. En este certificado existirán diferentes campos que nos informarán sobre aspectos como la dirección IP, la autoridad de certificación, etc. Estos campos se pueden clasificar según la información que contienen:

Información relativa a la identidad del receptor:

- Dirección IP.
- Número de interfaces asociados a los dispositivos.

- Direcciones MAC.
- Clave pública.
- Privilegios.

Información relativa al contexto:

- Identificador de red.
- Nivel de seguridad.
- Modo de acceso.
- Parámetros de acceso.

Información sobre el algoritmo de encriptación utilizado:

- Identificador del algoritmo.
- Lista de parámetros.

Información sobre la autoridad de certificación:

- Tipo de certificado.
- Servidor del certificado.

Información sobre la velidez del certificado:

- Tickets

Por último, tendríamos la firma del certificado.

BLOQUE III

10. Conclusiones y trabajo futuro

Una vez finalizado el trabajo, en este capítulo vamos a proceder a valorar los resultados. En conjunto, creemos que hemos superado con creces las expectativas que teníamos al comienzo del proyecto.

La temática del trabajo ha sido nueva para nosotros tres, por lo que al principio fue difícil determinar si los objetivos marcados serían razonablemente alcanzables. Debido a nuestro desconocimiento de los campos en los que hemos trabajado, el proyecto ha resultado tener una componente muy importante de documentación y estudio de otros trabajos previos, incluyendo artículos científicos altamente técnicos.

A pesar de esta falta de conocimientos previos, a lo largo del curso hemos adquirido una sólida base que nos ha permitido no sólo entender las soluciones existentes actualmente en los campos de investigación de redes MANET y seguridad de redes; sino también razonar sus puntos fuertes, sus defectos, y proponer soluciones que podrían integrarse algún día en protocolos usados en entornos reales.

Por supuesto, entendemos que el proyecto resultante no es perfecto. Somos conscientes de los defectos y carencias que sufre nuestro protocolo SARA, y de que sería posible mejorar en muchos aspectos.

En concreto, de haber dispuesto de más tiempo nos hubiese gustado plantear un sistema de comprobación para el mecanismo de sincronización. En este momento, el protocolo que presentamos no tiene manera de detectar errores en la sincronización de las tablas *Free_IP_Blocks*, ni forma de corregir esos posibles errores. Queda pues como una sugerencia para trabajos futuros que quieran tomar como base nuestra memoria el desarrollar este sistema.

Nuestra propuesta de incluir en los mensajes OLSR información sobre la autoconfiguración sería una forma de comprobar los bloques de direcciones IP libres que cada nodo tiene asignados, por lo que nosotros consideramos que sobre esa idea podría construirse este sistema de comprobación mencionado.

Otra funcionalidad que nuestro protocolo necesitaría para poder ser considerado completo y usado en entornos reales es la posibilidad de detectar y solucionar los problemas que se derivan de la unión de redes MANET. Esta funcionalidad fue considerada como poco prioritaria por nuestro director del proyecto, para poder dedicar más tiempo a desarrollar el módulo de seguridad del protocolo. Por ello, desconocemos hasta qué punto sería difícil incluir en el protocolo SARA esta funcionalidad, pero lo consideramos un reto interesante que podría ser objeto de estudio en un trabajo futuro.

También esperamos que el módulo de seguridad implementado sea algún día ampliado para ofrecer seguridad a las redes MANET. Creemos que al proporcionar en este proyecto una base que facilita la inicialización de la red y una gestión básica de la seguridad, es un buen punto de partida para desarrollar algún sistema de seguridad avanzado para este tipo de redes. En redes MANET existen multitud de retos de cara a la seguridad, siendo quizás el que más nos ha llamado la atención conseguir crear un entorno de monitorización y gestión de confianza de manera distribuida.

11. Referencias:

[1] James A. Freebersyser, Barry Leiner, “A DoD perspectiva on mobile ad hoc networks”, ddison Wesley, Reading, MA, pp. 29-51, 2001.

[2] Sushant Jain, “Energy Aware Communication in Ad – Hoc Networks”, Technical Report, University of Washington, Seattle, Junio 2003.

[3] E. Baccelli, INRIA, "Internet Draft: Address Autoconfiguration for MANET: Terminology and Problem Statement", February 2008

[4] S. Thomson, Bellcore, T. Narten, IBM, "RFC2462: IPv6 Stateless Address Autoconfiguration", http://www.ietf.org/rfc/rfc2462.txt 1998

[5] T. Narten, IBM, E. Nordmark, Sun Microsystems, W. Simpson, Daydreamer, "RFC2461: Neighbor Discovery for IP version 6 (IPv6)", http://www.ietf.org/rfc/rfc2461.txt 1998

[6] Troan, O., R. Droms, and Cisco Systems “IPv6 Prefix Options for DHCPv6,” Internet Engineering Task Force (IETF) http://www.ietf.org/rfc/rfc3633.txt, 2003.

[7] R. Droms, Ed., Cisco, J. Bound, Hewlett Packard, B. Volz, Ericsson, T. Lemon, Nominum, C. Perkins, Nokia Research Center, M. Carney, Sun Microsystems "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)",http://www.isi.edu/in-notes/rfc3315.txt July 2003

[8] S. Nesargi and R. Prakash, “MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network" Proc. IEEE INFOCOM 2002, New York, NY, June 2002.

[9] F. Ros, P. Ruiz, University of Murcia, C. Perkins , Nokia Research Center, “Internet Draft: Extensible MANET Auto-configuration Protocol (EMAP)” October 2005

[10] C. Perkins, Nokia Research Center, E. Belding-Royer, University of California, Santa Barbara, S. Das, University of Cincinnati, “Ad hoc On-Demand Distance Vector (AODV) Routing” http://tools.ietf.org/html/rfc3561 July 2003

[11] Mansoor Mohsin and Ravi Prakash, The University of Texas at Dallas, Richardson, “Ip address assingment in a mobile ad hoc network”, 2002

[12] C. Perkins Charles Perkins, Jari Malinen, Ryuji Wakikawa, Yuan Sun and Elizabeth M. Belding-Royer, “IP Address Autoconfiguration for Ad Hoc Networks,” IETF draft, 2001.

[13] N. H. Vaidya, “Weak Duplicate Address Detection in Mobile Ad Hoc Networks,” Proc. ACM MobiHoc 2002, Lausanne, Switzerland, June 2002, pp. 206–16.

[14] Kilian Weniger, Institute of Telematics University of Karlsruhe, “Passive Duplicate Address Detection in Mobile Ad Hoc Networks”, Germany

[15] Longjiang Li, Yunze Cai, Xiaoming Xu, Yonggang Li, “Agent-Based Passive Autoconfiguration for Large Scale MANETs”, July 2007

[16] Namhoon Kim, Soyeon Ahn, Younghee Lee, “AROD: An address autoconfiguration with address reservation and optimistic duplicated address detection for mobile ad hoc networks” 2007

[17] Y. Sun and E. M. Belding-Royer, “Dynamic Address Configuration in Mobile Ad Hoc Networks,” UCSB tech. rep. 2003-11, Santa Barbara, CA, June 2003.

[18] Kilian Weniger, Universit¨at Karlsruhe (TH), “PACMAN: Passive Autoconfiguration for Mobile Ad hoc Networks” Germany

[19] Basagni Stefano, Conti Marco, Giordano Silvia, Stojmenovic Ivan. 2004. “Mobile Ad hoc Networking”. Wiley Inter-Science. 1st Edition. New Jersey. 461 pp.

[20] Geetha Jayakumar, G. Gopinath, “Ad Hoc Mobile Wireless Networks Routing Protocols – A Review”, Bharathidasan University, Journal of Computer Science 3, 2007

[21] Charles E. Perkins, Pravin Bhagwat, “Highly Dynamic Destination-Sequence Distance-Vector Routing (DSDV) for Mobile Computers”

[22] I. Chakeres, Motorota, C. Perkins, “Internet Draft: Dynamic MANET On-demand (DYMO) Routing”, http://ianchak.com/dymo/draft-ietf-manet-dymo-12.txt, February 2008

[23] D. Jonson, Rice University, Y. Hu, UIUC, D. Maltz, Microsoft Research, “The Dynamic Source Routing Protocol (DSR)for Mobile Ad Hoc Networks for IPv4” http://www.ietf.org/rfc/rfc4728.txt February 2007

[24] V. Park, S. Corson, Flarion Technologies, Inc., “Internet Draft: Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification”, July 2001

[25] C. Hedrick,Rutgers University "Routing Information Protocol" http://tools.ietf.org/html/rfc1058, June 1998

[26] T. Clausen, Ed., P. Jacquet, Ed., Project Hipercom, INRIA, “Optimized Link State Routing Protocol (OLSR)” http://www.ietf.org/rfc/rfc3626.txt, October 2003

[27] R. Ogier, SRI Internacional, F. Templin, Nokia, M. Lewis, SRI Internacional, “Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)” http://www.ietf.org/rfc/rfc3684.txt February 2004

[28] Mario Gerla, Xiaoyan Hong, Guangyu Pei, Rockwell Scientific Company,"Internet Draft:Fisheye State Routing Protocol (FSR) for Ad Hoc Networks", June 2002

[29] Zygmunt J. Haas, Marc R. Pearlman, Prince Samar, “Internet Draft: The Zone Routing Protocol (ZRP) for Ad Hoc Networks”, July 2002

[30] John M. McQuillan, Isaac Richer and Eric C. Rosen, *ARPANet Routing Algorithm Improvements*, BBN Report No. 3803, Cambridge, April 1978

[31] Daniel Lang, “On the Evaluation and Classification of Routing Protocols for Mobile Ad Hoc Networks”

[32] Farooq Anjum, Petros Mouchtaris "Security for wireless ad hoc networks"

[33] The Handbook of ad hoc wireless networks, Chapter 30:Security in ad hoc networks - Amitabh Mishra, Ketan M. Nadkarni, Virginia Polytechnic Institute and State University

[34] RSA Laboratories, ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf, June 2002

[35] El Gamal, T. “Cryptography and logarithms over finite fields”, tesis de PhD, Stanford University, 1984.

[36] Diffie, W. y M.E.Hellman, "New directions in cryptography", IEEE Transactions on Information Theory 22 (1976), pp. 644-654.

[37] Joseph M. Miret Biosca, “Criptografía con curvas elípticas”, Departamento de matemáticas, Universidad de LLeida

[38] Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman. “NTRU: A Ring Based Public Key Cryptosystem”, http://www.ntru.com/cryptolab/pdf/ANTS97.pdf , June 1998

[39] R. Housley, SPYRUS, W. Ford, VeriSign, W. Polo, NIST, D. Solo, Citicorp, “Internet X.509 Public Key Infrastructure Certificate and CRL Profile”, http://www.ietf.org/rfc/rfc2459.txt, January 1999

[40] Manel Guerrero Zapata, Mobile Networks Laboratory Nokia Research Center FIN-00045 NOKIA GROUP "Secure ad hoc on-demand distance vector routing"

[41] Lenstra, A. K. and Lenstra Jr., H. W. (editors), “The development of the number field sieve”, 1993

[42] S. Pohlig and M. Hellman. "An Improved Algorithm for Computing Logarithms over GF (p) and its Cryptographic Significance”, *IEEE Transactions on Information Theory* (1978), pp. 106–110.

[43] Agrawal, M., Kayal, N. and Saxena, N. "Primes is in P." http://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf, 2004

[44] Weisstein, Eric W. "Rabin-Miller Strong Pseudoprime Test." From MathWorld A Wolfram Web Resource. http://mathworld.wolfram.com/Rabin-MillerStrongPseudoprimeTest.html

[45] The NS-3 Network Simulator, http://www.nsnam.org/

[46] http://es.wikipedia.org/wiki/Test_de_primalidad

[47] http://es.wikipedia.org/wiki/Hash

[48] R. Rivest, MIT Laboratory for Computer Science, and RSA Data Security Inc.,"The MD5 Message-Digest Algorithm" http://www.ietf.org/rfc/rfc1321.txt, April 1992

12. Anexos

12.1. NS-3

El simulador NS-3 (*Network Simulator 3* [45]) está diseñado dentro del proyecto NSNAM (*Network Simulator Network ANimator*) siendo un simulador de redes de eventos discretos.

Este simulador está escrito en el lenguaje C++, y se ofrece bajo la versión 2 de la GPL (*GNU General Public License*).

Las principales características de este simulador de redes son:

- Fácilmente extensible
- Código abierto y libre para el estudio o modificaciones
- Multiplataforma
- Desarrollo en comunidad

12.1.1. Protocolos y estructuras implementadas

NS-3 tiene implementadas diferentes capas del modelo TCP/IP, y en ellas se encuentran diferentes protocolos y estructuras de datos contempladas en los protocolos de redes actuales.

Al ser un simulador que se desarrolla en comunidad y con código abierto, las implementaciones existentes han podido aumentar gracias a desarrolladores de todo el mundo.

La figura 12.1 muestra una tabla que resume las estructuras y protocolos ya implementados oficialmente tanto en NS-3, como en su antecesor NS-2:

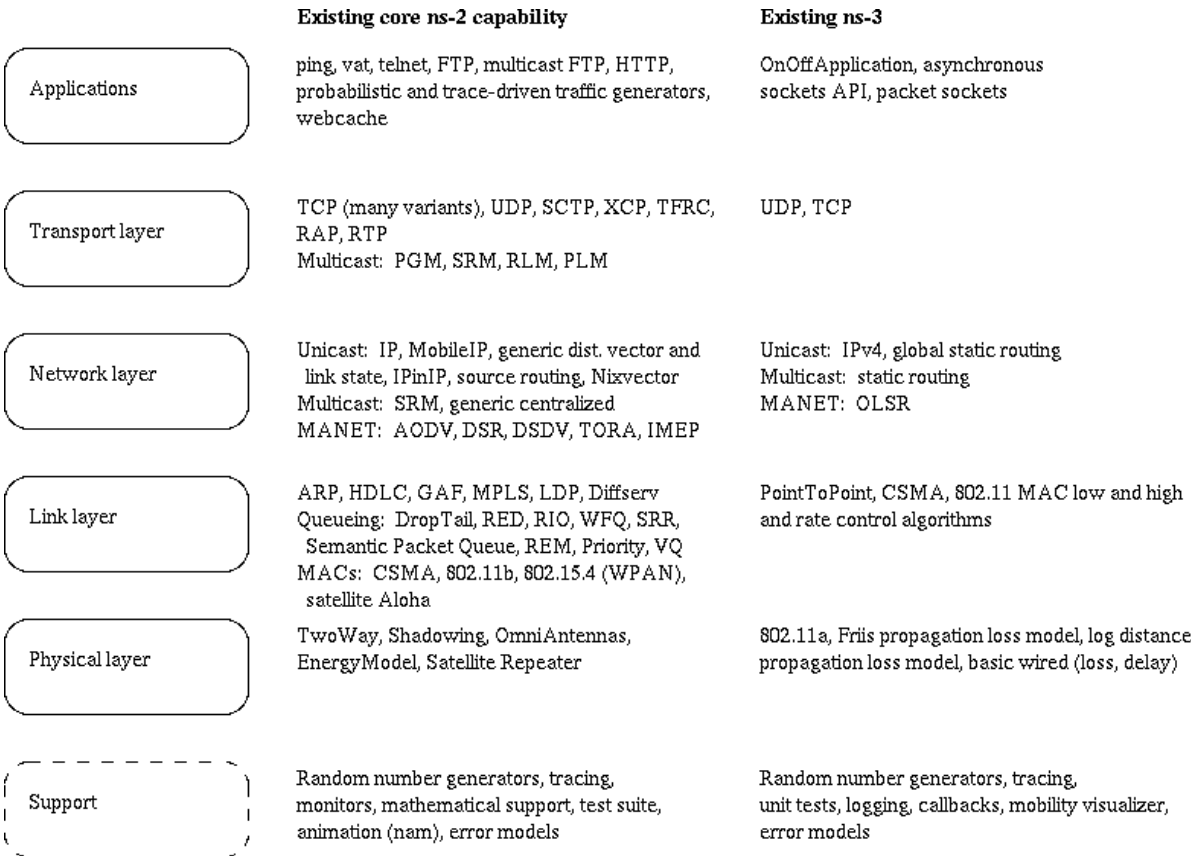


Fig. 12.1 – Tabla resumen de estructuras y protocolos implementados en NS-2 y NS-3

12.1.2. Escenarios e implementación

Para la generación de escenarios se implementan *Scripts* escritos en C++ y Python. Estos escenarios cuentan con determinados objetos, llamados entidades (*entities*) que son utilizados de forma análoga a objetos del mundo real, como son *sockets*, aplicaciones o canales.

Las entidades mínimas que se deben programar en los escenarios son:

- **Nodos:** forman la red, se conectan a través de los canales de transmisión.
- **Aplicaciones:** simulan el envío y recepción de datos en los nodos.
- **Paquetes:** objetos que se envían, por ejemplo, un datagrama UDP.
- **Protocolos:** se encuentran entre la aplicación y el interfaz de red. Simulan conexiones, acceso a un medio compartido, direccionamiento, encaminamiento, etc.

- **Interfaces de red:** es la interfaz entre el protocolo y el canal de transmisión, de esta manera se implementan Ethernet, CSMA (*Carrier Sense Multiple Access*), PPP (*Point to Point Protocol*), etc.

- **Canales:** Modelan el medio de transmisión, introduciendo determinados parámetros como retardos, tasa de transmisión, etc. Estos canales unen los nodos.

12.1.3. Diagrama de las capas del protocolo TCP/IP

La figura 12.2 gráfico muestra el tránsito de información a través de las diferentes capas del modelo TCP/IP representado con las entidades del simulador NS-3.

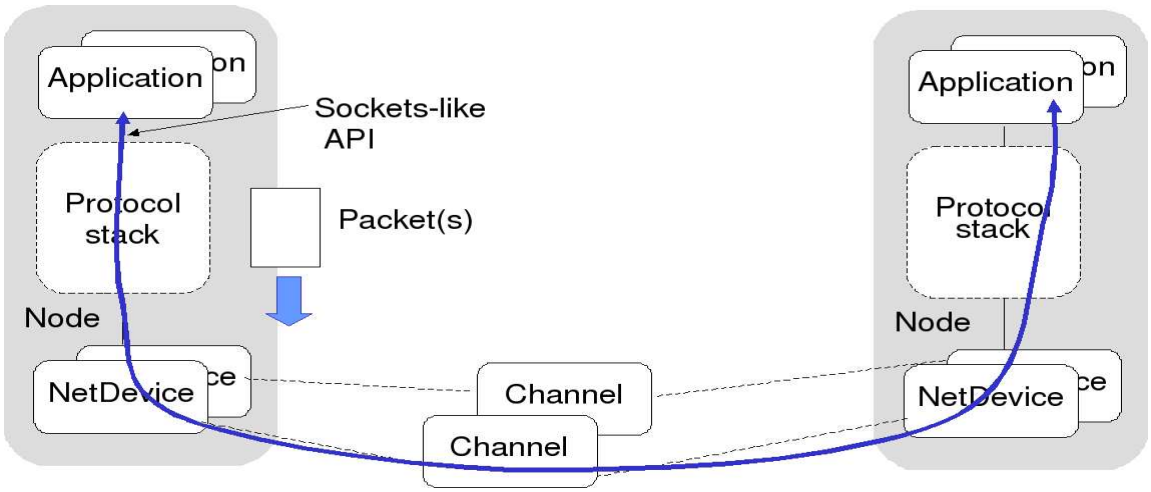


Fig. 12.2 – Tránsito de información entre capas TCP/IP de NS-3

12.1.4. Ejemplo de un escenario

El siguiente *Script* es un ejemplo que viene dado por los desarrolladores para la comprensión del sistema de entidades. Este *Script* llamado *simple.cc* muestra cómo crear nodos, y asignarle diferentes configuraciones.

Primero crea un nodo al que simula una conexión a Internet y un intercambio de mensajes, para después mostrar el tráfico procesado por el *socket* simulado:

- En primera instancia se debe crear un *NodeContainer* que será el contenedor que tendrá los nodos de la simulación, en este caso creamos 1 nodo.

- El siguiente paso es la asignación de los diferentes *Helpers.* Los *Helpers* sirven para la creación de los diferentes protocolos, y estructuras de datos dentro de las redes de comunicación. De esta manera, los *Helpers* pueden considerarse como un grupo de clases que implementan las diferentes configuraciones aplicables a un nodo, a un contenedor o a un dispositivo. Por ejemplo, en este caso se le dota al nodo con el protocolo necesario para simular Internet, mediante el *InternetStackHelper*. Análogamente se pueden asignar un *Helper* de encaminamiento como *OLSR-helper* a los diferentes nodos, o un *Helper* que implementa las direcciones IP versión 4 (*ipv4-address-helper*).

- Por último queda lanzar la ejecución con los parámetros deseados.

Código del ejemplo:

```
Void RunSimulation (void)
{
 NodeContainer c;
 c.Create (1);
 InternetStackHelper internet;
 internet.Install (c);
 TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
 Ptr<Socket> sink = Socket::CreateSocket (c.Get (0), tid);
 InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), 80);
 sink->Bind (local);
 Ptr<Socket> source = Socket::CreateSocket (c.Get (0), tid);
 InetSocketAddress remote = InetSocketAddress (Ipv4Address::GetLoopback
        (), 80);
 source->Connect (remote);
 GenerateTraffic (source, 500);
 PrintTraffic (sink);
 Simulator::Run ();
 Simulator::Destroy ();
}
```

12.1.5. Estructura de clases

El siguiente esquema resume el sistema de módulos disponibles en el simulador NS-3. Esta estructura de clases facilita la tarea del programador, ya que es fácil encontrar los elementos deseados para una simulación, así como las configuraciones requeridas para la estructura de la red recreada. Estos son los principales módulos:

- **Simulator:** se centran las clases involucradas en la simulación en sí, como son *Time* y *Scheduler*. Objetos de estas clases son necesarias para cualquier simulación en un escenario del simulador NS-3, ya que se usan para simular todos los eventos que se suceden en una simulación, como la entrada o salida de nodos de una red.

- **Node:** encontramos todas las clases usadas para la configuración de nodos, como la dirección IP, el canal de transmisión, los diferentes dispositivos de red o *sockets***.**

- **Devices:** modelan diferentes accesos a un medio de transmisión, como un puente (*bridge*), el modelado del protocolo punto a punto (PPP), el protocolo CSMA o el medio WIFI**.**

- **InternetStack:** se centra en la implementación de los protocolos necesarios para Internet, como son TCP, UDP o ARP.

- **Helpers:** se engloban todos los *Helpers* creados hasta el momento por el equipo NSNAM. Encontramos *Helpers* destinados a aplicar diferentes configuraciones a los dispositivos de la red como InternetStackHelper; pero también encontramos otros que son útiles para crear un conjunto de objetos con una propiedad en común como CsmaHelper (el cual crea dispositivos capacitados con el protocolo CSMA).

- **Applications:** en este módulo existen un conjunto de clases que simulan el comportamiento de la capa de aplicación del modelo TCP/IP, como son los mensajes *UDP Echo*.

- **Mobility:** se describen una serie de modelados de movilidad, tales como la posición en ejes cartesianos, la velocidad o la dirección que van tomando diferentes nodos.

- **Routing:** corresponde con los protocolos de encaminamiento implementados en este simulador. Hasta el momento están implementados OLSR y el encaminamiento global.

12.2. Implementación y codificación

En esta sección trata de mostrar diversas codificaciones sobre diversos algoritmos que nos han surgido a la hora de implementar el protocolo.

12.2.1. Generación de números primos: algoritmo Miller-Rabin

No existe ningún algoritmo que genere de forma directa números primos en un intervalo. Pero existen algoritmos que generan números aleatorios dentro de un intervalo de los que posteriormente se comprobará su primalidad.

Para verificar la primalidad de un número, existen dos modos: realizar un test de primalidad o una prueba de primalidad.

- **Test de primalidad**: es un algoritmo que, dado un número de entrada *n*, no consigue verificar la hipótesis de un teorema cuya conclusión es que *n* es compuesto. Es decir, conjetura que "ante la falta de certificación sobre la hipótesis de que *n* es compuesto, podemos tener cierta confianza de que se trata de un número primo". Supone un grado menor de confianza que lo que se denomina una prueba de primalidad (o test verdadero de primalidad) que ofrece una seguridad matemática al respecto.
- **Prueba de primalidad**: es un algoritmo determinístico que, dado un número de entrada *n*, verifica la hipótesis de un teorema cuya conclusión es que *n* es primo. Una prueba de primalidad es la verificación computacional de dicho teorema.

Así pues, se puede hablar de dos grados de incertidumbre: las pruebas de primalidad (existe certidumbre matemática) y los tests de primalidad (existe certidumbre práctica).

El test de primalidad más implantado es el *Test de Miller-Rabin*[44], pero en 2002 se presento la *Prueba de primalidad AKS*[43], que es un algoritmo determinista de orden polinómico. Sin embargo, tiene la desventaja con *Miller-Rabin* de que es bastante más lento.

Debido a que *AKS* es un algoritmo bastante más lento y *Miller-Rabin* tiene la garantía de ser el más usado, nos decantamos por éste para la generación de números primos necesaria para los algoritmos del criptosistema.

El *Test de Miller-Rabin*[44] es un test de primalidad que decide si un número es o no primo.

El algoritmo que sigue es el siguiente:

- Entrada: *n* impar, con $n=s^2 \cdot r +1$
- Elegir: $1 < a < n$ al azar. Si:
  - $a^r \equiv 1 \pmod{n}$
  - $a^{2^{(j)}r} \equiv -1 \pmod{n}$, $0 \leq j \leq s-1$, entonces *n* es primo con una probabilidad mayor o igual que 3/4. En otro caso, *n* es compuesto.
- Aplicar el test *k* veces aumenta la probabilidad a $1 - 1/4^k$.

El *Test de Miller-Rabin* tiene un orden de complejidad $O(k \cdot log^3 n)$ siendo *k* el número de ejecuciones que se realiza el test determinándonos su fiabilidad.

Una posible codificación del algoritmo se muestra en la figura 12.3 [46]:

Entrada: Un número natural *n*>1, el número *k* de veces que se ejecuta el test y nos determina la fiabilidad del test.

Salida: COMPUESTO si *n* es compuesto y POSIBLE PRIMO si *n* es un posible primo.

1. **Definase** r y s tal que r es impar y $(n-1) = r \cdot 2^s$
2. **Para** j desde 1 hasta k **haga lo siguiente:**
    1. $a \leftarrow$ **Función** Genera_numero_aleatorio_en_intervalo$[2, n-2]$
    2. $y \leftarrow a^r \bmod n$
    3. **Si** $(y \neq 1) \wedge (y \neq n-1)$ **entonces:**
        1. $j \leftarrow 1$
        2. **Mientras** $j \leq (s-1) \wedge y \neq (n-1)$ **haga lo siguiente:**
            1. $y \leftarrow y^2 \bmod n$
            2. **Si** $y = 1$ **entonces:**
                1. **Retorne** COMPUESTO
            3. $j \leftarrow j+1$
        3. **Si** $y \neq (n-1)$ **entonces:**
            1. **Retorne** COMPUESTO
3. **Retorne** POSIBLE PRIMO

Fig. 12.3 – Posible codificación del algoritmo Miller- Rabin

12.2.2. Funciones *hash:* MD5

Una función de *hash* es una función para resumir o identificar probabilísticamente un gran conjunto de información, dando como resultado un conjunto imagen finito generalmente menor (un subconjunto de los números naturales por ejemplo). Varían en los conjuntos de partida y de llegada y en cómo afectan a la salida similitudes o patrones de la entrada. Una propiedad fundamental del *hashing* es que si dos resultados de una misma función son diferentes, entonces las dos entradas que generaron dichos resultados también lo son [47].

Un *hash* es el resultado de dicha función. Son usadas en múltiples aplicaciones, como los arrays asociativos, criptografía, procesamiento de datos y firmas digitales, entre otros. Una buena función de *hash* es una que experimenta pocas colisiones en el conjunto esperado de entrada; es decir que se podrán identificar unívocamente las entradas.

Existen multitud de algoritmos que implementan esta función, los más conocidos son: AES, MD4, MD5, SHA, SHA-1, Tiger, WHIRLPOOL o RIPEMD-160.

En http://www.cryptopp.com/benchmarks.html se puede consultar una tabla con diferentes características de varios algoritmos existentes en la actualidad.

En nuestro protocolo, la función *hash* utilizada es MD5. Fue diseñado por Ronald Rivest en 1991 en el Instituto Tecnológico de Massachussets.

Se creó MD5 (*Message-Digest Algorithm 5*) [48], siendo una extensión de su versión anterior MD4, después de que se descubriera su debilidad. MD5 es ligeramente más lento que MD4 pero es más seguro ante posibles ataques.

Obtiene resultados de 128 bits, mucho más cortos que otros algoritmos *hash* habitualmente representados como cadenas de 32 dígitos en formato hexadecimal.

En los últimos años se ha puesto en entredicho su futuro, después de que en agosto de 2004 descubrieran colisiones de *hash* en sus resultados.

El tamaño del *hash* de 128 bits, se considera demasiado corto frente a ataques de fuerza bruta, con los que se cree que podrá ser reventado.

El funcionamiento del algoritmo lo podemos encontrar en su RFC: http://www.ietf.org/rfc/rfc1321.txt.